UNIVERSITY OF TWENTE.

Faculty of Behavioural, management and Social Sciences

Testing the application of quantile regression in screening insurance claims by assigning conditional quantiles

Laurens H. K. Schipper

M.Sc. Thesis

Febuary 2022

Conducted at:

Posthuma Partners

Supervisors:
dr. C.G.M. Groothuis-Oudshoorn
dr. B. Roorda
M. Nijmeijer (external member)

Faculty of Behavioural, management and Social Sciences,
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Management summary

Screening insurance claims for errors or fraud is an important aspect of the work of insurance companies. It is important for these companies that they detect wrong insurance claims and correct them in order to prevent overpaying. Rather than fully inspect each claim, usually insurance companies make a preselection with a yearly target rate, where they investigate a certain percentage of all claims further, commonly around 30%. This can be done through multiple means, for example inspecting all claims above a predetermined value or, what Posthuma Partners proposes, is, by using regression methods, assigning all claim amounts a conditional quantile and investigating every claim with an assigned quantile above a threshold. The current models have inherent assumptions about the distribution of claim amounts, which affects how the models assign quantiles. One of the models used is ordinary least squares regression. This model has two important assumptions that do not always hold true: homoskedasticity and normally distributed errors. These assumptions contribute to inconsistent quantile assignment, which results in an inconsistent preselection process. In the past, Posthuma Partners has developed a model that is able to work with log-linear variance, a form of heteroskedasticity: LMVAR. However, the model still assumes that the errors follow a normal distribution. It is common that using these methods, in order to achieve the target ratio during the year the number of preselected quantiles needs to be increased or decreased. This causes operational problems, since it is unclear how much work needs to be done.

In order to reduce these problems, a different type of regression is analysed in this research: Quantile regression. Rather than first estimating the mean, assuming the error is normally distributed and then calculating the quantile from that distribution, quantile regression directly estimates the conditional quantile, removing the need for the normality assumption. This leads to the main research question:

How can quantile regression be used to improve conditional quantile assignment for insurance claim amounts?

In order to see how well quantile regression really performs, first a synthetic dataset was created, where the number of observations, the number of features, and the distribution of the dependent variable, or claim amount, are all changed. The design of this dataset was such that it resembles a real dataset in number of observations, number of features, and distribution of the dependent variable. On this dataset, quantile regression is tested to see how well it is able to correctly identify the true relation between the features and the claim amount and the true conditional quantiles. Rather than created a single quantile regression model for only one quantile, it is important for Posthuma Partners that the entire distribution is known. For this reason, a set of 39 quantile regression models is used in this phase, where each model estimates one quantile $(0.025\%, 0.05\%, \ldots, 0.975\%)$. From the combination of these models, almost the entire distribution is known. Using this information, the R^2 is calculated in order to be able to compare it to the traditional regression models and the quantile distribution is checked for uniformity using the Kolmogorov–Smirnov test (KS test).

The results of this initial testing on the synthetic dataset show that quantile regression is both able to assign the conditional quantiles accurately and uniformly under certain conditions. The minimum number of observations at which the model performs adequately seems to be 10,000, which most of the available datasets surpass. Another condition is that if the claim amounts follow a distribution with non-linear variance, this greatly impacts the performance as well. If it is the case that the non-linearity is too impactful, the quantile regression models are merely able to estimate if the claim amount is higher or lower than the median.

After the initial testing, further research begins on a real dataset of car insurance claims. Firstly, this dataset is analysed. The number of observations, the distribution of the claim amount and other important aspects are explained. After that, the features and what they represent are discovered. Some observations needed cleaning, for example the car brand was not capitalised, or there was data missing.

The next step is to create three models: a OLS-regression model, an LVMAR model, and a combined quantile regression model. These models are trained on a subset of all insurance claims available in the dataset. The goal is to be able to compare the performance of these three models and see if quantile regression is a viable option. The models are compared mainly in three ways, the first of which is how well the β coefficients are estimated and how these are impacted by a feature selection method. OLS-regression and quantile regression both use

LASSO as a feature selection method, and LVMAR uses stepAIC with backward-forward selection. The second comparison is the R^2 and some additional extensions on the R^2 to better represent overfitting. Thirdly, the conditional quantile distribution is analysed and compared. This is done using the Kolmogorov-Smirnov test.

The results show that the in general the models perform about equally with respect to the R^2. The OLS-regression model performed the best with an R^2 of 0.5, followed by the combined quantile regression model with an R^2 of 0.486, and finally the LMVAR with an R^2 of 0.470. The feature selection methods did remove many features from the model, without impacting the R^2 significantly for both the OLS-regression model and for the LMVAR model. However, the quantile regression model performed much worse, with an R^2 of 0.255. It is likely that this bad performance is due to the fact that the quantile regression model is a combined model, and that the feature selection model is distorting it.

The distribution assigned quantiles is supposed to be uniform. However, both the OLS-regression and the LMVAR models were not able to do this uniformly. The results from the KS test show that for both distributions, the null-hypothesis of uniformity can reasonably be discarded, where LMVAR was only marginally better than the OLS model. The quantile regression model was able to pass the KS test, with a P-value of 0.3095. This means that it can reasonably be assumed that the assigned conditional quantiles from the quantile regression model follow a uniform distribution.

The conclusion of this research is that quantile regression is applicable on datasets that are similar to the datasets that are available to Posthuma partners. The distribution of the assigned quantiles that quantile regression provides is very consistent, which might reduce variability in workload during the year. Since quantile regression performs worse on the datasets with non-linear variance, it is advisable to test with the LMVAR model as well. The minimum number of observations is around 10000.

Based on the analysis of the available data, there are a number of improvements. The input of the dataset can be constraint more, such that data is more standardised. This would result in a more usable dataset with more information. This applies for example very well to the car brand feature. A feature that would likely be worth adding is a season related feature. Currently, this is not used, since the data is only available for one year in the past. By analysing data over a longer timespan, seasonal effects can be measured as well. Another improvement would be to update the models that are used more frequently. The current model performance was not very well, likely due to the effects of COVID-19 on the car insurance industry. However, these effects can be reduced by updating the models monthly, rather than yearly.

The quantile regression model need more research as well. The feature selection method LASSO performed very poorly, which needs to be investigated further. Furthermore, there are more applications of quantile regression that can be used as well: quantile neural networks, quantile decision trees, and random forests that use quantile regression. These models can also predict conditional quantiles, with possibly better performance than just a regression model. Another advancement in quantile regression is Additive Smoothing. This type of model can be researched further as well.

Preface

With much pleasure, I present you this thesis, which I have written as my graduation project for the master Industrial Engineering & Management, with as specialization Financial Engineering & Management at the company Posthuma Partners. The focus of this thesis is on exploring the application of quantile regression in screening insurance claims by assigning conditional quantiles. The goal of this application is to assign higher quantiles to claims that are more expensive than similar claims, such that insurers can more easily identify those claims. During the writing of this thesis, I learned a lot about statistical learning, data mining, data processing and visualization, but also about how to conceptualize a problem, and how to think about the bigger picture of a problem. In short, I learnt a lot and could not have done it without the support from the people around me.

Firstly, I would like to thank Marco Nijmeijer for the many online meetings we have held and the continuous availability of help when it was needed. Even though, due to the well known Corona pandemic, we did not meet in person during the entire period I was working on the thesis, your help provided me with a lot of insights into writing this thesis. I would also like to thank the other colleagues I spoke to online.

Furthermore, I would like to thank Karin Groothuis-Oudshoorn. Our regular meetings helped me a lot, and you always had great advice. Next, I would like to thank Berend Roorda, my second supervisor, for the additional advice and proofreading.

Lastly, I would like to give a big thank you to my girlfriend, my family, and friends. Finalizing this thesis marks the end of my student life, which I enjoyed very much, but I am even more excited by what is yet to come!

Yours sincerely,
Koen Schipper
Febuary 2022

Contents

Management summary ii

Preface iv

List of Figures vii

1 Introduction 1
- 1.1 Posthuma Partners' core activities . . . 1
- 1.2 The Claims Management Filter . . . 1
- 1.3 Likelihood of claims . . . 3
- 1.4 Linear regression . . . 4
  - 1.4.1 Linear regression with log-linear variance . . . 5
- 1.5 Quantile regression . . . 5
- 1.6 Main problem statement . . . 5
- 1.7 Research objective . . . 6
- 1.8 Research questions . . . 6

2 Quantile regression 8
- 2.1 Quantile regression . . . 8
  - 2.1.1 Finding the optimal β's with linear programming . . . 8
  - 2.1.2 Penalized linear quantile regression . . . 10

3 Performance measurement 11
- 3.1 Conditional distribution estimation . . . 11
  - 3.1.1 Conditional mean estimation for a univariate stochast . . . 11
  - 3.1.2 Multivariate stochast . . . 12
  - 3.1.3 Conditional variance beta . . . 12
- 3.2 Uniformity of quantile distribution . . . 12
- 3.3 Flagging claims . . . 12
  - 3.3.1 Flag rates . . . 13

4 Model Design for testing on synthetic data 14
- 4.1 Synthetic datasets . . . 14
  - 4.1.1 Parameters of the synthetic datasets . . . 14
  - 4.1.2 Performance measurement on synthetic datasets . . . 16

5 Results on synthetic data 18
- 5.1 Number of observations . . . 18
- 5.2 Features . . . 20
- 5.3 Type of dependent variable generation method . . . 22

6 Preliminary analysis of available data 24
- 6.1 Datasets . . . 24
- 6.2 Features . . . 24
  - 6.2.1 Car related features . . . 25
  - 6.2.2 Accident related features . . . 25
  - 6.2.3 Repair related features . . . 26
  - 6.2.4 Insurance related features . . . 27
  - 6.2.5 Costs . . . 27
  - 6.2.6 Remaining dataset . . . 27

6.3 Current model performance . . . 28

7 Model Design for real data **29**
7.1 Regression models design . . . 29
7.1.1 OLS regression . . . 29
7.1.2 LMVAR . . . 29
7.1.3 Quantile regression . . . 29
7.2 Performance comparison methods . . . 30
7.2.1 Feature influence comparison . . . 30
7.2.2 R^2 with variants . . . 30
7.2.3 Quantile distribution . . . 31
7.2.4 Flag rates and Kappa . . . 31

8 Results on real data **32**
8.1 Model results . . . 32
8.1.1 OLS regression . . . 32
8.1.2 LMVAR . . . 33
8.1.3 Quantile regression . . . 33
8.2 Variance reduction . . . 35
8.3 Quantile estimation . . . 36
8.4 Flag rates and kappa's . . . 37
8.5 Overall results . . . 38

9 Conclusions and recommendations **39**
9.1 Conclusions . . . 39
9.2 Recommendations . . . 39
9.3 Discussion . . . 40

A List of available datasets **43**

B List of features **44**

C Code for synthetic datasets **45**
C.1 Creating the synthetic dataset . . . 45
C.2 Quantile regression on subsets of the synthetic dataset . . . 45
C.3 Quantile regression on different features and number of features . . . 50
C.4 Quantile regression on a synthetic dataset based on LMVAR . . . 52

D Code for synthetic datasets **54**
D.1 Cleaning the original dataset . . . 54
D.2 Code for applying quantile regression on the dataset . . . 55
D.3 Code for OLS regression . . . 60
D.4 Code for LMVAR models . . . 61

List of Figures

1.1 Flow chart for the statistical process of the CMF . . . 2
1.2 The flag rate over time . . . 3
1.3 The sample quantiles 0.05, 0.5, and 0.95 shown on the density plots for a standard normal distribution . . . 4
1.4 A histogram of the frequency that the quantiles occur with 10 bins and an added line that shows the average frequency . . . 6

2.1 Optimization view of different cost functions . . . 9

4.1 true conditional quantiles in an example of OLS regression . . . 16
4.2 true conditional quantiles in an example of LMVAR with $\beta_\sigma = 0.5$ . . . 16

5.1 Beta coefficient estimation for the intercept and for one variable . . . 19
5.2 Mean square error of beta estimation for multiple quantiles . . . 19
5.3 The relation between sample size and beta estimation for mean and variance . . . 20
5.4 Quantile estimates versus theoretical estimates . . . 20
5.5 RMSE over quantiles for different sample sizes . . . 21
5.6 Plot of sample size vs calculation times . . . 21
5.7 The relation between features and model performance . . . 21
5.8 The impact of the number of features on the calculation time . . . 22
5.9 The relation between features and model performance . . . 22

6.1 Box plot of the relation between type of vehicle and total costs . . . 25
6.2 Box plot of the relation between cause of accident code and total costs . . . 25
6.3 histogram of repair time . . . 27
6.4 histogram of number of rows . . . 27
6.5 Histogram of the logarithm of the total claim costs . . . 28
6.6 histogram of the assigned quantiles of the total cost of claims . . . 28

7.1 The swaps that occurred when sorting the conditional quantile conjugate predictions . . . 30

8.1 CV plot of the lambda's . . . 32
8.2 OLS LASSO beta values . . . 32
8.3 LMVAR mu beta values . . . 33
8.4 LMVAR sigma beta values . . . 33
8.5 Intercept beta coefficient over quantiles . . . 34
8.6 'Cause of accident 1' beta coefficient . . . 34
8.7 Density plot for the μ of the beta coefficients . . . 34
8.8 Density plot for the variance of the beta coefficients . . . 34
8.9 Cumulative Distribution Function of one observation . . . 35
8.10 Probability Density Function of one observation . . . 35
8.11 Histograms of the assigned quantiles for the OLS regression models . . . 36
8.12 Histograms of the assigned quantiles for the OLS regression models . . . 36
8.13 Histogram of assigned quantile of quantile regression . . . 37
8.14 Empirical distributions of the different models . . . 37
8.15 Flag rates over time for the three models at quantile 0.85 . . . 38

Chapter 1

Introduction

This research is performed at Posthuma Partners. Posthuma Partners is an actuarial consulting firm and software developer for insurance companies located in Gouda. Their customers are some of the largest insurance companies worldwide, like Achmea, Aegon and Allianz. They operate mainly in Northern Europe and Southeast Asia. This research aims to provide Posthuma Partners an additional method of data analysis and modelling.

In this chapter the research will be outlined. In section 1.1 the core activities of Posthuma Partners will be explained and which of these activities will be the focus of this thesis. After this the scientific context of this research will be given in chapter 2. Thirdly, the main problem statement will be described in chapter 1.6. From this problem statement, the research objective will be crafted and stated in chapter 1.7. This objective will be used to set out the research questions in chapter 1.8.

1.1 Posthuma Partners' core activities

The three main products and services Posthuma Partners provide are the following [16]:

- Integral financial modelling;
- Consultancy - Data analysis;
- Claims Management Filter (CMF).

The integral financial modelling is focused on providing the customer with stochastic modelling of their insurance portfolios and a strategic road map substantiated with quantitative metrics. The data analysis consultancy, which Posthuma Partners provides, gives the customers the opportunity to present the actuarial problems they face. Posthuma Partners is able to fulfil many roles, such as actuary, data analyst, and certifier of the portfolio. They can advise on pricing of products and can validate models and results. The third service is the implementation of the CMF. The CMF is a data analysis tool that can be used to aid in handling insurance claims. An insurance claim, or claim for short, is a formal request by an insured party to the insurance agency for (financial) compensation. An example of a claim would be if the car of an insurance policyholder is damaged in an accident. They can report the costs to repair the damage to their insurer, who will then, if the claim is deemed legitimate by the company, reimburse the customer. The claim contains information about the incident plus the requested claim amount. It is the job of the insurance company to verify the claims, such that they know the claim is valid, and the customer is entitled to the compensation. Then they can deny or validate the claim. The insurance company can have many thousands or even millions of customers, so the number of claims can be very high. For each of the claims, the insurer has to have experts inspect the claims who have to decide the likelihood that the claim is legitimate and that the claim amount is reasonable. The CMF can automate a large part of the verification of claims and make a preselection for further investigation. The company claims that the ratio of paid out claims is 2% to 20% lower with the product plus a reduction of claims handling costs of 20% to 50%.[17]

1.2 The Claims Management Filter

The CMF aids in claim verification in two parts: automation of the handling of claims and by providing expert advice for insurance claims, which can be one of three things:

- Green light: Routine claim processing
- Yellow light: further investigation via a phone call

- Red light: further investigation via an on-site expert

The ratio of claims that are flagged depends on the preference of the insurance company. In Europe, it usually is around 70% or more of the claims get green lights. Between 10% and 30% of the claims get a yellow light and around 10% get red lights. However, in Asian regions, this statistical method is not used at all, since the customers of Posthuma Partners prefer to further investigate all claims. In figure 1.1 the process is visualized.

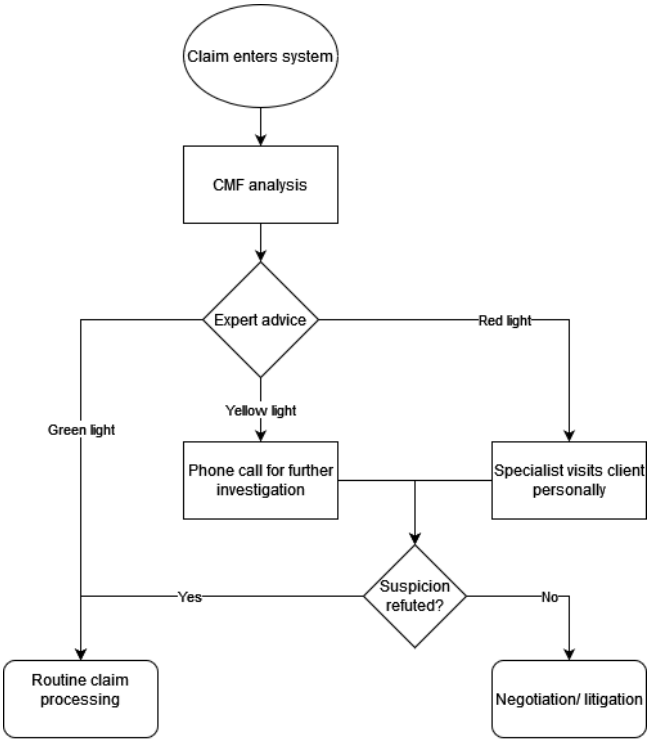


Figure 1.1: Flow chart for the statistical process of the CMF

Rather than randomly checking a portion of the claims, the CMF analysis bases the advice on three parameters. The first is a rule-based analysis of the incoming insurance claims. An example of this would be: 'The individual issuing the claim needs to be insured for the damage they report.' Rules can also be more aimed at checking the validity of the claim itself. Such as in the case of car paint damage, whether the colour of the paint is one that occurs on that type of car (there may be no red Ford Fiesta's, for example). The rules are determined by the insurer and implemented into the tool in cooperation with Posthuma Partners. Based on the outcomes of these rules, a score is granted between 0 and 1. Overall, this score should be uniformly distributed among all claims.

The second mean is a statistical method to make a preselection of the claims that should be further investigated. Currently, the norm for the method of choosing the claims to investigate is taking a slightly arbitrary monetary value above which all claims must be checked, for example 2000 euro. Of course, not all expensive claims are unreasonable and not all non-expensive claims are reasonable. For example, if two expensive cars collide, the claim amount will be high. In other words: the claim does not deserve to be flagged as unreasonable simply because it is high. The other way around works as well. If a certain procedure that normally can be done for a very low price, it should be alarming to receive a claim for such a procedure with a high claim amount. For this reason, CMF also provides a statistical analysis of the claims. Using regression models, the claims are given scores based on their likelihood.

The third parameter is based on the number of claims that have been further investigated and how many claims there are expected to be made. As mentioned before, insurance companies want a certain percentage of claims to be checked per year. It is undesirable that there are too few claims checked, but also too many. One of the main customers of Posthuma Partners is a expert agency in car insurance, which has multiple insurance companies as their customers. The contract between these insurers and the expert agency is that a predetermined percentage of the claims has to be checked per year, with monetary fines if the number of checked claims is too low. If the expert agency checks too many claims, the costs of checking those redundant claims are wasted. The agency is therefore incentivized, to aim at the exact right ratio of checked claims. Periodically, for example monthly, the agency calculates the current ratio and makes a prediction in how many claims are yet to come. Based on these data, the flag rates are calculated. At the start of the year, the flag rate is set at

the predetermined flag rate. If at some point during the year, the ratio of flagged claims is too low, the future flag rate, F_f, will be set above the predetermined flag rate F_p, such that with the expected influx of claims the total flag ratio over the entire year should result to the predetermined flag rate. If the ratio of flagged claims is too high, the opposite occurs.

$$F_f = \frac{F_p(N_h + N_f) - N_{hf}}{N_f} \tag{1.1}$$

Where F_f is the future flag rate and F_p is the predetermined flag rate. N_h, N_f, and N_{hf} are respectively the number of historical claims, (expected) future claims and the flagged historical claims. This results in a system where during the year, the strictness of the system changes, based on the performance of the model. Figure 1.2 shows the stacked area chart of the flag rates per month for all the expert agencies claims of September 2020 till August 2021. Green is of course for green flags, yellow for yellow and red for red. This is calculated on a dataset of twelve months, starting at September. The contracts between the expert agency and its customers are yearly from January to December. While this is a rather short time frame, it is visible that at the end of the calendar year the flag rate dropped. This is because the expert agency has flagged too many claims during the year and to compensate the future flag rate is lowered. The expert agency prefers a more stable flagrate, since this creates a more predictable workflow for the company and the likelihood of a claim being investigated is less dependent on the timestamp of the claim. Since this flagrate is partly determined by the statistical model, this will be investigated.

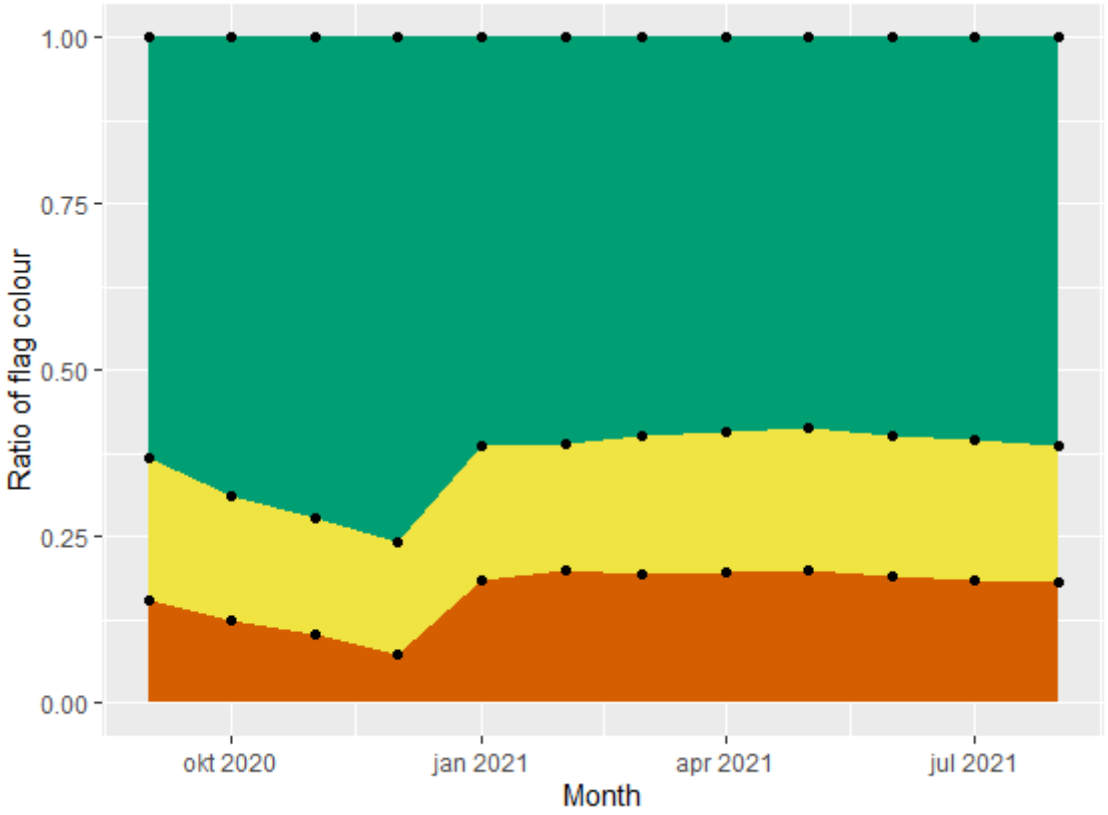


Figure 1.2: The flag rate over time

1.3 Likelihood of claims

The statistical model is designed to assign scores to claims based on their likelihood. This is done by using historical claims to calculate a likely distribution of costs conditional on the claim itself. Based on this conditional distribution, the claim can be given a score based on the conditional quantile the claimed costs in. A quantile is defined as follows:

Definition 1.3.1. (Quantile) Let Y be a real valued random variable with the cumulative distribution function (CDF) $F_Y(y) = P(Y \leq y)$. For all $0 \leq \tau \leq 1$, the τ-quantile of Y is given by:

$$Q_Y(\tau) = F_Y^{-1}(y) = inf\{y : F_Y(y) \geq \tau\} \tag{1.2}$$

The value of Y associated with a quantile τ will be referred to as the quantile conjugate in this thesis. For the case of a claim, the quantile conjugate would be the claimed costs. Following the definition of quantiles, the frequency of the quantile conjugate values is distributed uniformly between 0 and 1. This characteristic also means that the quantile function is monotonically increasing as the quantile conjugate increases. This is unaffected by transformation of the quantile conjugate, as long as the transformation does not change the order of the values.

As an example of quantiles, take the distribution of a standard normal distribution In this case the quantiles 0.05, 0.5, and 0.95 can be represented as follows:

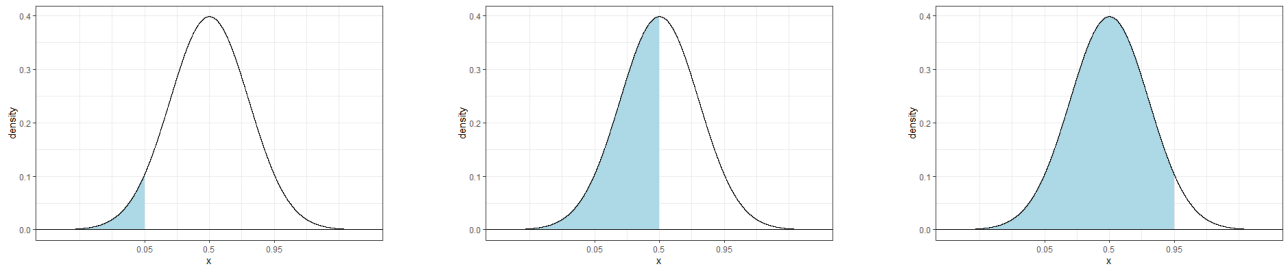


Figure 1.3: The sample quantiles 0.05, 0.5, and 0.95 shown on the density plots for a standard normal distribution

1.4 Linear regression

The conditional quantile of a claim can be calculated by different means. The easiest one is by taking the unconditional quantile: The ratio of values that are lower versus the total number of values. This approach leaves room for improvement. As was mentioned in the previous section, not all insurance claims are alike. Some damages are inherently more costly than others. For this reason, a conditional quantile is used [15]. Conditional quantiles are commonly calculated via linear regression. The idea of linear regression is to model a linear relationship between the response variable, also known as the dependent variable, and various explanatory, or independent, variables. In this paper, the word for explanatory variables used will be 'features'. The quantitative values of the features are represented by the following vector: $X^T = (X_1, X_2, ..., X_P)$, where P represents the number of features. The model uses a number of observations, n. Each observation is in this paper an insurance claim, where the dependent variable is the claimed cost. The features are details about the claim, for example which type of car is damaged or the location of the damage.

The relationship is estimated using a linear model, so it can be written as follows:

$$Y = \beta_0 + \sum_{p=1}^{P} \beta_p X_p + \epsilon \tag{1.3}$$

$Y = (Y_1, ..., Y_N)$ is the response variable in this case, with all N elements being a separate observation or claim. β_0 is the base value the model takes. If the values of all other features are 0, the estimated amount would be this β_0. The other β's represent weights of the values of the features. By taking the dot product of these β's and the feature values, a prediction can be made with some uncertainty ϵ.

To improve the readability of the function, it is common to add a $X_0 = 1$ to X. With this and creating a vector $\beta^T = (\beta_0, \beta_1, ..., \beta_N)$, formula 1.3 can be written as:

$$Y = X^T\beta + \epsilon \tag{1.4}$$

While linear regression assumes linear relationships between the features and the response variable, it is possible to use a transformation of the features or the response variable to find different kinds of relations, such as logarithmic, exponential, or quadratic.

The β's are estimated using a loss function. The goal is to pick β's such that the loss function has a minimal outcome. Depending on the loss function, the model could regress to the mean, median, or a predetermined quantile. The loss functions are respectively the mean squared deviation (MSE), mean absolute deviation (MAE), and the quantile check function. The most common regression, Ordinary Least Squares regression (OLS regression), is a regression on the mean, meaning that the estimation is assumed to be the average, or mean, of the full distribution of possible values. Thus, the loss function to minimize is the MSE:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(Y_n - X_n^T\beta)^2 \tag{1.5}$$

Minimizing this cost function finds the optimal β's under the condition that the errors are homoskedastic.

Definition 1.4.1. (Homoskedasticity) Let ϵ_i be the error of the i'th observation, with $E(\epsilon_i) = 0$. Under Homoskedasticity the following is true for all i:

$$E(\epsilon_i^2) = \sigma^2 \tag{1.6}$$

By adding another assumption, the assumption of normally distributed errors, Y can be described as a stochastic variable $Y_i \sim N(X_i^T\beta, \sigma^2)$. Here σ^2 is the variance of the error. With these two assumptions, the conditional quantile τ of a claim amount can be calculated with the following formula, where Y_τ is the value of the $P(Y \leq Y_\tau) = \tau$:

$$Y_\tau = X^T\beta + \sigma\phi(\tau) \tag{1.7}$$

In formula 1.7, the $\phi(\tau)$ is the standard normal cumulative distribution (CDF) value of quantile τ. To extract the quantile, the inverse of this distribution is used.

1.4.1 Linear regression with log-linear variance

The assumption of homoskedasticity does not always hold. In the case of insurance claims, it is entirely possible that the variance of claim amounts is different for different sizes of claims amounts. This is also what Posthuma Partners had noticed, and for this reason they use a different model for some datasets: the LMVAR model. This model is still linear in the expectation value. However, it assumes heteroskedasticity, the opposite of homoskedasticity, in a log-linear form. This means that it is assumed that there is an extra set β's and X's such that $log(\sigma) = X_\sigma\beta_\sigma$. This changes the stochastic variable Y to $Y_i \sim N(X_i^T\beta, e^{2X_\sigma^T\beta_\sigma})$. Calculating the quantile of such a variable can be done with the following function:

$$Y_\tau = X_\mu^T\beta_\mu + e^{X_\sigma^T\beta_\sigma}\phi(\tau) \tag{1.8}$$

Both regression methods can assign quantiles to claims, by estimating a mean and variance for each claim. Using these parameters, the quantile that represents the claim amount is calculated for that specific normal distribution.

1.5 Quantile regression

In this thesis, a different method of regression is applied. One that does not rely on the Gaussian distribution of the errors. The method is quantile regression. Quantile regression is a form of linear regression, but rather than first estimating the mean and variance of an observation, it directly estimates a linear relation between a quantile and its quantile conjugate. Quantile regression was first introduced by Koenker et al in 1978[4]. In the paper, it is shown how generalizing the loss function for median regression (the MAE), can be used to create quantile regression. This function is called the check function. It allows for quantile regression by letting the user pick a quantile τ and perform linear regression on that quantile, by penalizing positive errors with a factor τ and negative errors with a factor $-(1-\tau)$. In the case of the median τ would be 0.5. The check function can be formulated as:

$$\rho_\tau(\epsilon) = \begin{cases} \tau\epsilon & for\ \epsilon > 0 \\ -(1-\tau)\epsilon & otherwise \end{cases} \tag{1.9}$$

This paper laid the foundation of quantile regression. Since then there have been many additions, like feature selection, cross-validation and others.[11]

1.6 Main problem statement

As mentioned before, Posthuma Partners scores insurance claims partly based on the perceived quantile they are in. While Posthuma Partners have good models for many of the datasets they analyse, there are some hurdles in assigning claims the correct quantiles. In appendix A, an overview of datasets on which Posthuma Partner says their model underperforms according to their standards is given. They judge the performance of their models on a broad range of criteria. The most important among these are the R^2, quantile distribution, p-values of KS-tests, mean absolute errors, and an analysis of significant features.

However, for the datasets in appendix A only two measures are given: distribution of quantiles and the R^2. The dataset "Company E Total Costs*" scores poorly on both these criteria and will therefore be the initial focus of this thesis. The first criterion is the distribution of quantiles. In figure 1.4 the distribution of quantiles is shown in a histogram with 10 bins. By definition, quantiles should all have a similar frequency. When the histogram is compared to the average frequency, it is clear that this distribution does not seem uniformly distributed.

Another metric that is shown in the table is the R^2. This metric is the fraction of the total variance that can be described by the model.

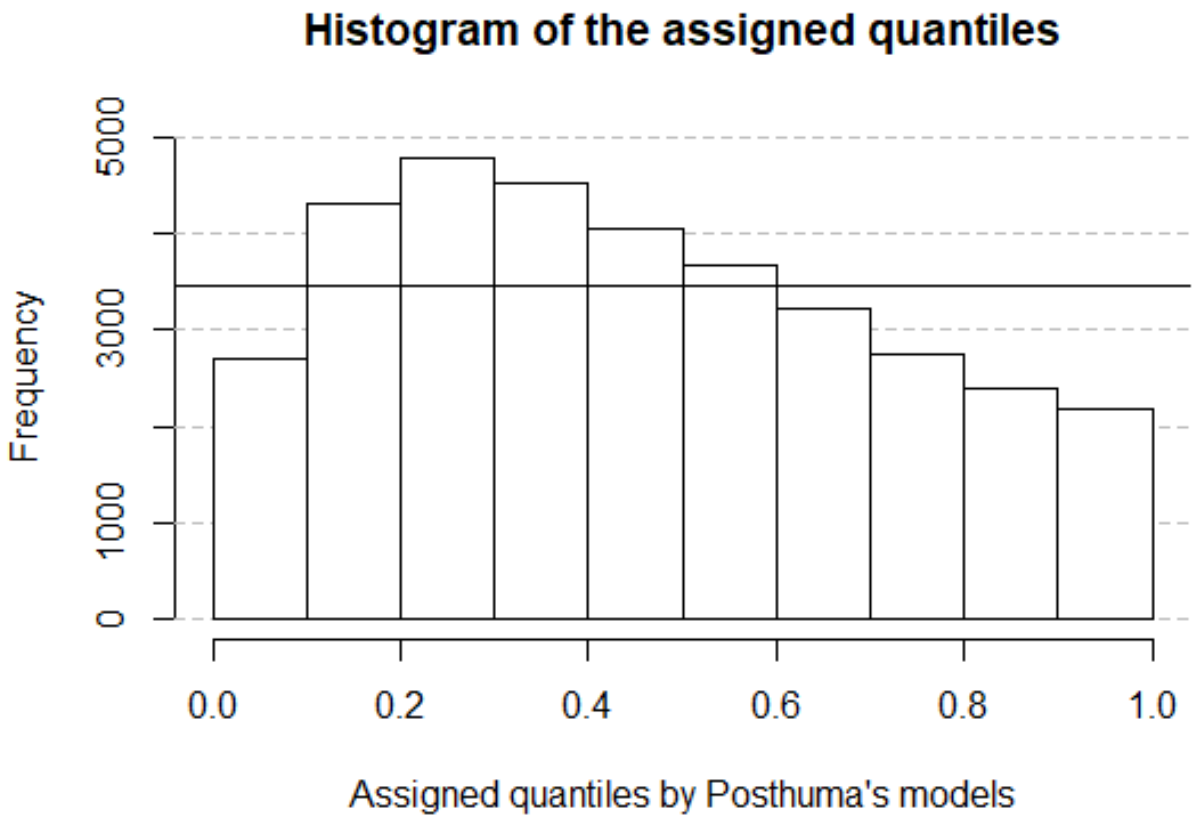


Figure 1.4: A histogram of the frequency that the quantiles occur with 10 bins and an added line that shows the average frequency

Definition 1.6.1. (R^2)

- Let SS_{res} be the sum of squares of the residuals of the model
- Let SS_{tot} be the sum of squares of the difference of values to the sample mean
- $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

The model used for the dataset in this case had an R^2 of 0.49. This means that about half of the variance is still unexplained. Some available datasets had a R^2 value of less than 0.49, but in combination with the bad quantile distribution, this dataset was deemed a good starting point for this research.

The problem statement is essentially that the models that are currently used, do not perform as desired on some datasets.

1.7 Research objective

The initial problem statement is that the statistical models do not assign uniformly distributed and accurate quantiles to insurance claims on some datasets. This can present itself in a worse quantile distribution or a lower R^2. The idea is that the normality assumption underlying both the OLS regression and the LMVAR regression models might be false and that rather than relying on these assumptions it may be better to go straight to quantile regression. The objective of this research is to explore the possibilities of quantile regression and design a model on a dataset that does a better job at assigning quantiles to claims.

1.8 Research questions

The main research question of this thesis is as follows:

How can quantile regression be used to improve conditional quantile assignment for insurance claim amounts?

In order to complete this research systematically, this main research question will be split up into multiple sub questions. The first sub questions are about the possibilities of quantile regression. These are mostly technical and are about improving the performance of the model.

1. How does quantile regression work?
2. How can variable selection be implemented in a quantile regression model?
3. How can quantile regression be used to model the mean and variance of a distribution?
4. How can performance of such a model be measured?

After a good technical foundation is created, the data will be explored and the rest of the research will be set up. This starts with exploring quantile regression in a simulated environment.

5. Which parameters of a dataset influence the performance of quantile regression?
6. How do these parameters impact the ability of quantile regression to accurately estimate the correct beta coefficients?
7. How do these parameters impact the ability of quantile regression to accurately estimate the theoretical quantile of the dependent variable?
8. How effective are these quantile estimates to calculate the conditional mean and variance?
9. How do these parameters impact the calculation times?

With the knowledge gained from testing in the simulated environment, the modelling on the real dataset can begin.

10. What are the properties of the dataset?
11. How do the current model perform on this dataset?
12. How does quantile regression perform on this dataset?

With the results of the performance of quantile regression on the real dataset, it is time to compare the results of quantile regression with that of the old models. With this comparison, an evaluation of the viability of quantile regression can be made.

13. Is quantile regression a viable alternative to the current models?

With this last question answered, the main research question can be answered. The goal is to show how quantile regression can best be implemented.

Chapter 2

Quantile regression

In the introduction, quantile regression and the quantile check function were shortly discussed. In this chapter, the basis of what quantile regression is and how it can effectively be used to model quantiles will be laid out. Furthermore, we will also describe how feature selection can be done with quantile regression.

2.1 Quantile regression

Quantile regression as mentioned before uses the quantile check function to estimate quantiles of data points of response variable. It does not require assumptions of the distribution of the error, since it directly models individual quantiles by solving an optimization problem. The data is used to attempt to find optimal β's that can predict the quantile τ of the range of possible outcomes for an observation. To do this the following optimization is established by Koenker [4]:

$$\min_{\beta \in \mathbb{R}}\Big(\sum_{t \in \{t:y_t \geq x_t\beta\}} \tau|y_t - x_t\beta| + \sum_{t \in \{t:y_t < x_t\beta\}} (1-\tau)|y_t - x_t\beta|\Big) \tag{2.1}$$

With the check function in formula 1.9 the general formula 2.1 can be simplified to the following:

$$\min_{\beta \in \mathbb{R}}(\sum_{t=1}^{n} \rho_\tau(y_t - x\beta)) \tag{2.2}$$

In this formula τ is the given quantile. At the median, or $\tau = 0.5$, this optimization is equal to optimizing the MAE. For other quantiles, the weights are asymmetric relative to the chosen quantile. The positive errors are given a weighting of τ and the negative errors are given a weighting of $-(1-\tau)$. Since negative and positive errors are penalized differently in such way, the β's create a linear model that represents the τ'th quantile for each observation. This means that in $100\tau\%$ of cases the actual quantile conjugate is below the predicted value.

2.1.1 Finding the optimal β's with linear programming

The question now becomes: How does the optimization of the β work? As mentioned before, there are a number of cost functions that can be minimized for different results. In his book, Quantile Regression (2005), Koenker shows a intuitive visualization, shown in figure 2.1. There are graphs that show the optimization view of the sample mean, median, and quantile τ. Finding the minimal value is usually a matter of setting the derivative of the formula to 0. With OLS regression or mean regression, this leads to the following formula: $\beta^* = (X^T X)^{-1} X^T Y$. This makes finding the optimal β's a matter of calculus. The problem with quantile regression is that, as can be seen in the graph, there is no derivative at the minimal point, since at that point there is no gradient.

For this reason, there needs to be a different approach on has to follow. The original approach described by Koenker, et al. [4] was to use linear programming, since this optimization is a combination of linear functions and restrictions. Linear programming problems are usually formulated in the following manner:

$$\min c^T z \tag{2.3}$$

Subject to:

$$b_i = a_i^T z, i = 1, ..., m, \tag{2.4}$$

$$z \in \mathbb{R}_+^n \tag{2.5}$$

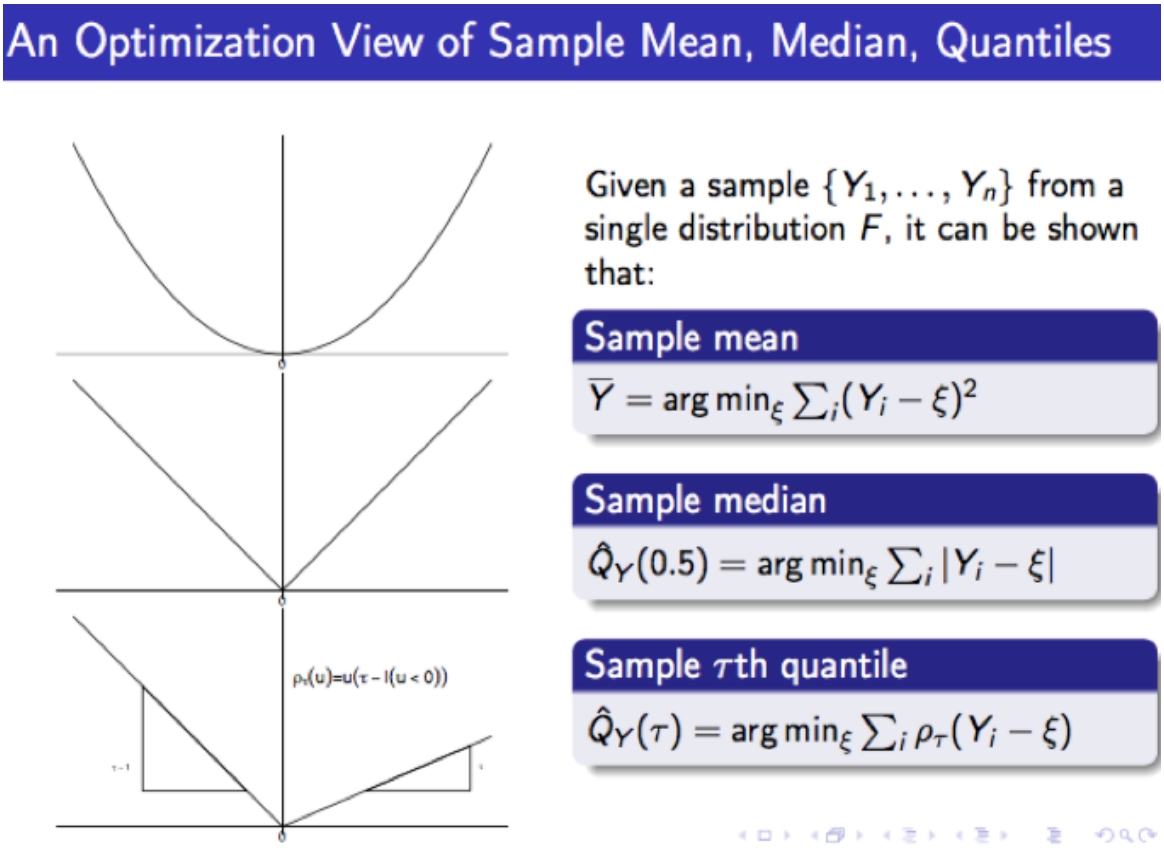


Figure 2.1: Optimization view of different cost functions

In quantile regression finding the optimal β's is done using the following formula:

$$\hat{\beta} := arg \min_{\beta \in \mathbb{R}^K} \sum_{i=1}^{N} \rho_\tau(y_i - x_i^T \beta) \tag{2.6}$$

Here $\hat{\beta}$ is the vector with length K of estimated optimal β's that minimize the formula. K is the number of features and N is the number of observations. This can be rewritten into the following linear program:

$$\min[\tau i' \epsilon^+ + (1-\tau) i' \epsilon^-] \tag{2.7}$$

Which is subject to:

$$\begin{gathered} Y = X(\beta^+ - \beta^-) + \epsilon^+ - \epsilon^- \\ \beta_\tau \in \mathbb{R} \\ \epsilon^+, \epsilon^- \in \mathbb{R}, \epsilon^+, \epsilon^- \geq 0 \end{gathered}$$

Here i' is a K vector of ones, with K being the number of features used in the model. Let ϵ^+ be $\max\{0, \epsilon\}$ and ϵ^- be $\max\{0, -\epsilon\}$. The same holds for β^+ and β^-

In order to rewrite this formula into the standard form, a few changes are made. First off, 'b' will be defined as Y. Secondly, the constraint will be rewritten as:

$$b = [X, -X, \mathbf{I}_N, -\mathbf{I}_N] \begin{bmatrix} \beta^+ \\ \beta^- \\ \epsilon^+ \\ \epsilon^- \end{bmatrix} \tag{2.8}$$

When $[X, -X, \mathbf{I}_N, -\mathbf{I}_N]$ is defined as $A^T = (a_1^T, ..., a_N^T)$ and $z^T = [\beta^+ \beta^- \epsilon^+ \epsilon^-]$, the constraint is rewritten to $b = A^T z$. This would make $c^T := [\mathbf{0}, \tau \mathbf{1}_N, (1-\tau)\mathbf{1}_N]$ with $\mathbf{0}$ being a $2K \times 1$ matrix of 0's. This would make the minimization problem:

$$\min(c^T z) = min(0\beta^+ - 0\beta^- + \tau i' \epsilon^+ + (1-\tau) i' \epsilon^-) \tag{2.9}$$

subject to:

$$b = A^T z$$

These types of linear program can be solved with simplex methods based on the median regression algorithms of Barrodale and Roberts[3]. This method is practical for datasets with up to a couple of thousand observations. This is due to computation times increasing rapidly.

For larger datasets, the Frisch-Newton algorithm is developed[7] to solve the linear programming problem. The algorithm uses interior point methods. According to the paper, the compututional speed is competitive with L_2-algorithms. L_2 refers to least-squares estimation models, in contrast to L_1 which refers to absolute error models. In chapter 4 and 5 the effect of these methods on calculation times is further investigated.

2.1.2 Penalized linear quantile regression

An important aspect of statistical modelling is feature selection. In this step, a methodology is chosen with which the most predictive features are chosen for the model. This can be done via different ways. However, the goal remains the same: decrease variability by increasing bias. A model with too few features can lose predictive value and exhibit a bias, but a model with too many features can be overfitted and thus exhibit a large variance. This can be done in different ways, one strategy is to alter the minimization problem to include a penalty for increasing the magnitude of the beta's. In order to reduce the number of features, the general formula from 2.2 can be adjusted as follows:

$$\min_{\beta \in \mathbb{R}^K} (\rho_\tau(Y - X^T\beta) + \lambda J(\beta)) \tag{2.10}$$

In this formula λ is the regularization parameter and $J(\beta)$ is the penalizing function. For the penalizing function J there are multiple options. In the paper 'Variable selection in quantile regression' [11] Yichao Wu and Yufeng Liu propose two methods of feature selection: Smoothly Clipper Absolute Deviation (SCAD) and Adaptive Least Absolute Shrinkage and Selection Operator (Adaptive LASSO) [8]. These two are proposed in this paper since these penalizers have a rate of convergence equal to $\sqrt{s/n}$, s is the number of features with a non-zero impact. This convergence rate is also known as the oracle convergence rate [14]. However, one of the issues of ALASSO is computation time, since it depends on multiple λ estimations. Belloni and Chernozhukov [14] developed a default value for λ. This method of lambda estimation is very quick in comparison to both the Adaptive and the SCAD methods, while remaining close to the oracle requirements of convergence.

LASSO

Regular LASSO, also called l1 regularization, is a method penalizing the regression by adding the absolute values of the β's. This results in the following formula:

$$\min_{\beta \in \mathbb{R}^K} (\rho_\tau(Y - X^T\beta) + \lambda \sum_{j=1}^{p} |\beta_j|) \tag{2.11}$$

LASSO has the nice property that the minimization problem remains a linear programming problem. Only the following alteration to the minimization problem needs to be made: c^T is changed to $[\lambda, -\lambda, \tau \mathbf{1}_N, (1-\tau)\mathbf{1}_N]$. The method remains Frisch-Newton. The choice for the λ is explained in the paper by Belloni and Chernozhukov [14]. The λ calculation is a formula based on the quantile, the standard deviation of the independent variables and the values of those variables.

StepAIC

A different method of feature selection is stepAIC. The idea of this method is to calculate the Akaike Information Criterion (AIC) which is a measurement describing the prediction error with relation to the number of features. A low AIC is generally better than a high AIC. Removing irrelevant features will decrease the AIC, while adding relevant features will also improve the score. This is exactly what the stepAIC method entails. Systematically, features will be added and removed to decrease the AIC. This functionality is available for the LMVAR model, however for quantile regression the LASSO method will be used. This is because the stepAIC function depends on a quick recalculation of the model, which is not the case for quantile regression.

Chapter 3

Performance measurement

As mentioned in the introduction, there are two important aspects of the models that are used: The ability to accurately estimate the distribution of the claim amounts and the ability to uniformly distribute quantiles over claims.

3.1 Conditional distribution estimation

The most common measure to assess the performance of regression models is the R^2 1.6.1. Comparing the performance of quantile regression with OLS-regression or LMVAR is not directly possible with the R^2. To circumvent this, the different conditional quantile predictions can be analysed together as a conditional distribution, with a conditional mean. This conditional mean can then be used to calculate the R^2.

3.1.1 Conditional mean estimation for a univariate stochast

The mean of this distribution will be extracted from the conditional distribution. This will firstly be done for a univariate stochast. The mean for a continuous univariate stochastic variable with probability density f_Y is defined like this:

$$\mu = \int_{y=-\infty}^{y=\infty} y f_Y(y) dy \tag{3.1}$$

The goal is to create a function for μ dependent on the quantile regression models for each τ. Therefore, we want to transform this integral over y to an integral over τ, where τ is a quantile ($0 \leq \tau \leq 1$) and $y(\tau)$ its quantile conjugate. We assume that $y(\tau)$ is a differentiable function with $\frac{\partial y}{\partial \tau} \geq 0$. This transformation is done by substituting for τ in formula 3.1, which results in:

$$\mu = \int_{\tau=0}^{\tau=1} y(\tau) f_Y(y(\tau)) \frac{\partial y}{\partial \tau} d\tau \tag{3.2}$$

Let f_τ be the PDF of τ. By the definition of quantiles, these are distributed uniformly between 0 and 1, with a constant density within this range of 1. Since the PDF of the quantiles is one to one related with the PDF of the quantile conjugates, these PDFs are as follows:

$$f_\tau(\tau) = f_Y(y(\tau)) \frac{\partial y}{\partial \tau} = 1, \text{ for } 0 \leq \tau \leq 1 \tag{3.3}$$

Using this, the formula of μ can be simplified to:

$$\mu = \int_{\tau=0}^{\tau=1} y(\tau) d\tau \tag{3.4}$$

Using quantile regression, a set of β's is obtained that can be used to model the quantile conjugates for each observation using the formula:

$$y(\tau) = X^\tau \beta_{QR}(\tau), \tag{3.5}$$

With $X \in \mathbb{R}^P$ the P-dimensional feature vector. Implementing this into formula 3.4 results in this:

$$\mu = \int_0^1 X^\tau \beta_{QR}(\tau) d\tau \tag{3.6}$$

In this formula X^T is a constant, so it can be placed outside the integral. Since all quantiles have the same likelihood and are evenly spread over the range from 0 to 1, the mean is the mean of beta estimates of each quantile regression model. This is a substitute of the integral and simplifies formula 3.6 into:

$$\mu = X^T \overline{\beta}_{QR}(\tau) \tag{3.7}$$

3.1.2 Multivariate stochast

The above analysis can be extended to multivariate stochasts relatively simple. In the case $Y \in \mathbb{R}^n$, where n would be the number of observations, the means $\mu \in \mathbb{R}^n$ are given by:

$$\mu = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} \tag{3.8}$$

For each observation i from 1 to n the following is true:

$$\begin{aligned} \mu_i &= \int_{-\infty}^{\infty} y_i f_Y(Y) dy_1, \cdots dy_n \\ &= \int_{-\infty}^{\infty} y_i f_i(y) dy \end{aligned} \tag{3.9}$$

Here y_i is the i'th element of Y and f_i is the marginal distribution of f_Y for y_i. Following the same arguments as for the univariate stochast, the elements of the vector μ can be written as follows:

$$\mu_i = x_i^T \overline{\beta}_{QR} \tag{3.10}$$

Here $x_i \in \mathbb{R}^P$ is the vector of feature values of y_i. The full vector μ can then be calculated by the following:

$$\mu = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} \overline{\beta}_{QR} = X \overline{\beta}_{QR} \tag{3.11}$$

What this means for this research is that if quantile regression can find sufficiently accurate quantile regression models, the regression on the mean can be estimated using those quantile models. Using this mean regression model, the R^2 of the combination of the quantile regression models can be calculated.

3.1.3 Conditional variance beta

Conditional variances can be calculated similarly as with the mean. Taking the population variance of the different quantile beta's of a feature results in a variance beta. However, it is important to note that these can not be summed up for a conditional variance. This is because of the covariance between the different features. The variance beta's do however give an indication of the effect the feature has on the skedasticity of the claim amount.

3.2 Uniformity of quantile distribution

In order to test the uniformity of the quantile distribution the Kolmogorov-Smirnov Test (KS Test) [1] will be used. With this test, it is possible to calculate the likelihood that a sample is drawn from a certain population. The quantiles should be drawn from a uniform population. A low p-value ($<= 0.05$) from the KS test means that H_0 can reasonably be discarded. In that case, it is unlikely that the sample is taken from the uniform distribution. The original KS method is computationally heavy, however in the paper Evaluating Kolmogorov's Distribution[9] a fast method is proposed to calculate the p-value with high precision. This method is also available in R.

3.3 Flagging claims

The third aspect of the models' performance is the quantile assignment to claims. If a claim has a certain quantile, the claim is flagged. The rate at which claims are flagged is called the flag rate. In order to compare the performance of different models, the flag rates need to be compared.

3.3.1 Flag rates

The goal of this research is to set up a new model, namely the quantile regression model. It is important to quantify how well the new model and the LMVAR are able to flag similar claims. Flagging claims creates a binary outcome, where the model either flags a claim, or does not. The models can be viewed as observers who individually and independently observe the data and categorize the observations. In the paper, Cohen and Jacob[2] proposed the metric "Cohen Kappa" or just "Kappa".

Definition 3.3.1. (Kappa or κ)

$$\kappa = \frac{\text{Probability-corrected observed agreement}}{\text{Probability-corrected potential agreement}} \tag{3.12}$$

The Kappa is a probability corrected form of measuring agreement between observers. The value is between -1 and 1, where 0 is no probability corrected agreement and 1 is full agreement. To give insight in how κ is calculated, view the following confusion matrix of two categories. In each cell of table 3.1 a letter represents the relative frequency of the combination of the outcomes of the two models. Since these values are relative frequencies, they are between 0 and 1.

		Model 1		
		True	False	Total
Model 2	True	A	B	(A+B)
	False	C	D	(C+D)
	Total	(A+C)	(B+D)	1

Table 3.1: Confusion matrix example

The probability corrected agreements can be calculated by first calculating the probability of random agreement. This is done by the following formula:

$$\text{Probability of random agreement} = (A+B) \times (A+C) + (C+D) \times (B+D) \tag{3.13}$$

With this probability, the observed agreement and potential agreement can be corrected. This is done in the case of observed agreement by subtracting the probability of random agreement. The probability corrected potential agreement is calculated by equation 3.15.

$$\text{Probability-corrected observed agreement} = A + D - \text{Probability of random agreement} \tag{3.14}$$

$$\text{Probability-corrected potential agreement} = 1 - \text{Probability of random agreement} \tag{3.15}$$

With these two values, κ can be calculated by dividing them. It follows from the formula's that if only B and C are 0, thus a complete agreement, the κ is 1. If only A and D are 0, the κ is between -1 and 0, depending on the probability of random agreement.

Chapter 4

Model Design for testing on synthetic data

After the literature research, the performance of quantile regression will be tested. This will be done using the following steps. Firstly, the performance of quantile regression will be tested in a synthetic environment. This way the impact of certain aspects can be analysed. To do this multiple datasets are created via a number of parameters. These are described in section 4.1.1. The performance will be quantified via a handful of measures that are explained in section 4.1.2. How the datasets are created is explained in section 4.1. The second step is to test quantile regression in a real life setting. With the knowledge of how quantile regression performs in different scenarios, it will be used on real data. This comprises on first translating the data that is available into one that can be used for regression and has all the important factors and features. Next is setting up the parameters along the models will be compared. After it is clear how the models will be compared, the models will be set up using the knowledge gathered in the literature review. The results will be discussed in the next chapter.

4.1 Synthetic datasets

In order to look at the performance of quantile regression, first it will be tested on a dataset of which the true parameters are known. This way, the impact of changing these parameters will be evaluated. Before the datasets are created, the metrics that will be used to measure the performance of the model will be set up. After that, the parameters will be discussed.

4.1.1 Parameters of the synthetic datasets

As mentioned above, there are multiple parameters that will be tuned to see the effect on the performance. The parameters are:

- Number of observations $[10^3, 10^{3.5}, 10^4, 10^{4.5}, 10^5]$
- Number of Boolean features [0 - 100]
- Number of numeric features [0 - 100]
- Beta's of the features for mean and variance [(-4 - +4); (-1 - +1)]
- Distribution of the dependent variable [standard linear model, LMVAR]

By changing these parameters, the goal is to find out in which circumstances quantile regression performs adequately. The ranges of the parameters are representative of the datasets that are available to Posthuma Partners. The dataset that is the focus of this dataset consists of around $10^{4.5}$ observations. The number of features is around 500 in the dataset. The beta's of the features when a LMVAR model is trained on the dataset range from -1.5 to +7 for the mu beta's and from -0.8 to +0.5 for the sigma beta's. How these parameters are tuned is explained in the following paragraphs.

The dataset will first be split into a training set of 80% of the observations and a test set of 20% of the observations.

Number of observations

By increasing the number of observations, the model has more data points that can be used to estimate the 'true' regression lines. This results in a more accurate model. The drawback of more observations is that the calculation time increases quickly for quantile regression. This is because, rather than with OLS regression, the method of calculating the beta's is a matter of linear programming. The set of number of observations will be created by taking increasing powers of 10, starting with 10^3, up to 10^5, with steps of 0.5. This set of options is chosen since the goal is to see changes relative to the number of observations. The difference between 1,000 and 10,000, is just as interesting as between 100,000 and 1,000,000.

First, the datasets with differing number of observations will be tested. After that, one number will be chosen based on the performance and calculation time. The quantile regression models will be created using the Quantreg package in R.

Number of Boolean and numeric features

As can be found in the description of the original dataset, there are three categories of features: Numeric, Boolean, and categorical. Categorical features are usually split up into dummy variables, for each category, one. These dummy variables have the value 'TRUE', if the observation is in that category, and 'FALSE' if it is not. Therefore, it can be said that a categorical feature is a set of Boolean features. This leaves just numeric and Boolean. It is important to find out if quantile regression is accurately able to utilise both data types. The synthetic dataset will therefore be created with a large number of Boolean features and a large number of numeric features. The Boolean features will be randomly assigned 1's and 0's with equal probability.

The numerical can have more values. The distribution can be many shapes, however for this testing environment, the numerical features will all have a standard normal distribution. This means that the mean is 0 and the standard deviation is 1.

In order to see the effect of the features, different number of Boolean and numeric features will be tested. When testing the effect of stepAIC, a number of independent variables is added as well, with a beta of 0. The goal is to see how increasing the complexity of the dataset will affect the performance and calculation time.

Distribution of the dependent variable and beta generation

In the real world a model merely attempts to imitate reality, however by generating the dependent variable using known models the performance of quantile regression can be accurately measured. The discussed models, OLS regression and the LMVAR model, assume normality in the dependent variable. This means that using the features as described in the previous paragraph and a list of beta's, a distribution for the dependent variable can be created.

In order to generate values according to the OLS regression method, there needs to be a beta value for each of the features and a standard deviation for the whole dataset. Using this, n dependent variable values can be generated using the normal distribution: $N_n(X\beta_\mu, \sigma^2 I)$, where I is the identity matrix with n rows. For all features, a β_μ is generated using a uniform distribution between -4 and +4. This way there is a wide range of both positive and negative beta's. The sigma is set to an arbitrary number. In this case, 2. Of course, there is also an intercept $\beta_{intercept}$. This is set to an arbitrary 2 as well. Since the dependent variable is normally distributed with a constant variance and the relation between the features, the relation between the quantiles of the distribution and the feature values should be linear as well. This is a result from the formula 1.7. In a simple example with one dependent variable Y and one feature X a number of conditional quantiles are visualized in figure 4.1. These quantiles are 0.9, 0.7, 0.5, 0.3, 0.1. As can be seen in the figure, the quantiles are parallel. This means that the beta for X is equal for all quantiles. The only difference is the intercept. The goal of this synthetic dataset is to find out if quantile regression is able to accurately find the beta's associated with the features and the intercepts when the dataset is generated based on a linear model.

The next method of generating the dependent variable will be based on the LMVAR model. The procedure will be similar as described above. However, the variance will be log-linear. This model needs two beta inputs for the features: β_μ and β_σ. The second value is necessary since the variance is dependent on the features as well. The β_σ values are taken randomly from a uniform distribution between -1 and +1. The distribution of the dependent variable is thus: $N(X\beta_\mu, e^{2X\beta_\sigma})$. Here $e^{2X\beta_\sigma}$ is a vector where each element is the exponent of the first row of X. It is important to note that it is not required that the X-matrix does not have to be identical for μ and σ. However, for the testing purposes both matrices were taken as equal.

Because the variance is increasing exponentially, the relation between the quantiles is no longer linear. How quantiles are calculated in a LMVAR model is described in formula 1.8. This creates a problem for quantile regression, since it assumes that quantiles are linear. An example of a simple dataset with only one feature is shown in figure 4.2. The quantiles do behave quite linearly at low X-values. Also the quantiles will behave more linearly when the β_σ is close to 0. At $\beta_\sigma = 0$ the variance of the model for that particular feature will always be 1, making it homoskedastic. The expectation is that quantile regression will perform worse on the lmvar

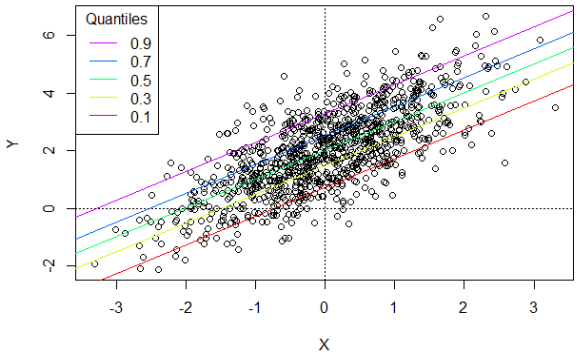


Figure 4.1: true conditional quantiles in an example of OLS regression

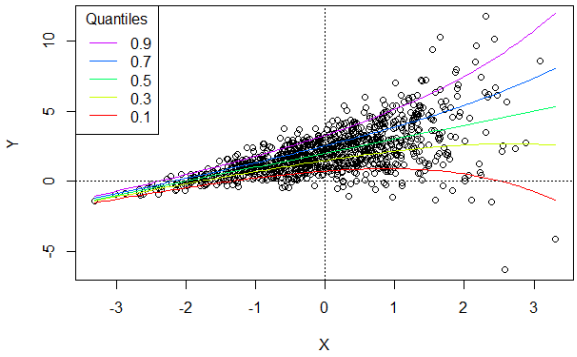


Figure 4.2: true conditional quantiles in an example of LMVAR with $\beta_\sigma = 0.5$

dataset than on the OLS regression dataset. However, as long as the β_σ's are close to zero enough, quantile regression will find reasonable results, since the relationship will be close to linear.

Unexplained variance

In all models, there is unexplained variance. This aspect will be chosen such that the ratio of the explained variance and unexplained variance is in line with the datasets that are available. This ratio is the best R^2 the model can theoretically find and will be set to around 0.67. This results in the unexplained variance being set at 4.

4.1.2 Performance measurement on synthetic datasets

Before the synthetic datasets are created and tested upon, the methods of performance measurement are established. These metrics are aimed at answering the following questions depending on the parameters of the dataset:

- How accurate are the beta's with relation to the theoretical values?
- How accurate are the mean beta and variance beta estimates based on these beta estimates?
- How well are the quantiles assigned to the claims?
- What are the calculation times?

In order to answer these questions, a quantitative measure is created for each question. These measures are obtained from literature as described in chapter 2.

Model performance

In the previous section, it is discussed how the dependent variable is generated. Using this knowledge, the model is compared to the generation process. The goal of this paragraph is to set up a method of quantifying the deviations.

Another measure to estimate how well the models estimate the generation process is to measure how close the beta estimates are, to the true beta's. As mentioned in section 4.1.1, the quantiles are known and can be estimated by quantile regression. It is interesting to see how the different parameters affect this performance.

The expected outcome for the linear model is that by increasing the number of observations and decreasing the complexity of the dataset, the estimated feature beta's will converge to the true values used to generate the dataset. The intercept beta depends on tau and the error distribution. Since the error distribution is known, the intercept beta can be calculated for each tau using the following formula: $\beta_{intercept}(\tau) = \phi^{-1}(\tau) \times \sigma + \beta intercept$. The true intercept and sigma values are known. These calculations are not possible with the LMVAR input model, since the quantiles are not linear. How well the estimated beta's fit the true beta's is quantified by calculating the MSE between the true and model beta's.

Quantile estimation

As mentioned before, the dependent variable is generated using random values taken from random variables with known parameters. Using this knowledge, the true quantiles for each observation is known.

By performing quantile regression multiple times on a range of different quantiles, a set of quantile conjugates for each observation is given. The next step is to compare the actual value of the dependent variable with these quantile conjugates and interpolate between the two closest values. The location of this interpolation is the estimated quantile.

The metric proposed here has the goal to quantify the discrepancy between the true quantiles and the estimated quantiles. For each observation, the true quantile value is subtracted from the estimated value. This just leaves the error. To see if there is a bias, the mean is taken from all errors. If this significantly deviates from zero, there is a bias in the model. Another measure is the standard deviation, also known as the root mean squared error (RMSE). This measures the spread of the errors. The reason that the RMSE is chosen, over the MAE, is to penalize variance. With MAE, all errors are given the same weighting, whereas with the RMSE the gives higher weighting to larger errors [13].

Since the practical application of quantile regression for this report is to identify claims in relatively high quantiles, it is important to see if the quantile estimation works well in those situations as well. Therefore, the observations are split into 20 bins based on the true quantiles. For each bin the measures mean and standard deviation are calculated again for the errors.

Mean and variance estimation

In the previous chapter, it was discussed how the mean and variance of a distribution can be calculated using quantiles. Since it is important to see if quantile regression can accurately describe the original data generation process, this will be done on the synthetic datasets as well. For each parameter change, the average difference between the $\hat{\mu}$ and the μ form of the generation formula will be calculated. Also, the difference between the standard deviation with the true standard deviation will be calculated. The reason for taking the square root is that the variance is a quadratic unit.

Calculation times

Quantile regression models are estimated using linear programming. This is a lot more time-consuming than using other types of regression. Seeing the impact of changing the different parameters on the calculation times is an important measure, since it gives an indication of the viability of the method. All calculations are done using the same computer and the same programming language, R. While it is true that calculation times depend on many factors, such as hardware, the relationship between the parameters and the calculation times can still give useful information.

Chapter 5

Results on synthetic data

In this chapter, the results of the research on quantile regression on a synthetic dataset will be discussed. The chapter will follow the same structure as chapter 4. For each parameter, the results of each metric is explained. Using these results, the setup of the modelling on the real dataset is fine-tuned. The effect of each parameter will be investigated in four aspects, as discussed in chapter 4:

- How accurate are the beta's with relation to the theoretical values?
- How accurate are the mean beta and variance beta estimates based on these beta estimates?
- How well are the quantiles assigned to the claims?
- What are the calculation times?

5.1 Number of observations

The first parameters was the number of observations. For these parameters a dataset is generated using 10 Boolean features and 10 numerical features, β's are randomly picked from -4 to +4, and a dependent variable based on a simple linear model with a standard deviation of 2. The number of observations are the following values: $[10^3, 10^{3.5}, 10^4, 10^{4.5}, 10^5]$. For each number of observations, 39 quantile regression models are run, for the quantiles 2.5%, 5%, ..., 97.5%.

Error in beta estimation

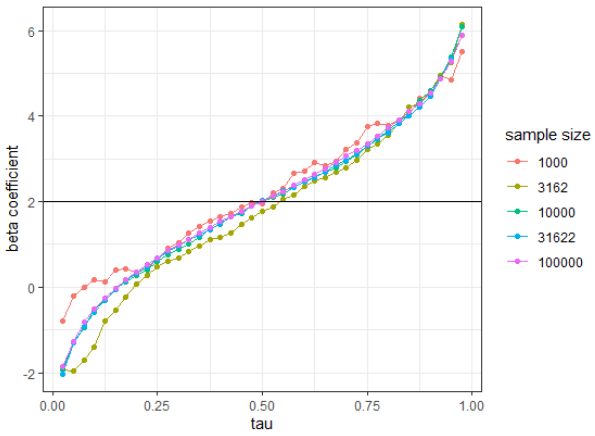
As mentioned in chapter 2, the beta's of a linear model can in theory be precisely approached with quantile regression with enough observations. Since the β's are known, the models β's can be compared to the true value. As an example, this is shown in figure 5.1b for feature 1, which was arbitrarily chosen. This figure shows how well the beta is estimated over the quantiles for several sample sizes. The black line is the true value from the generation process. The red line is the estimation for the smallest sample size: 1,000. It performs less predictably than the other sample sizes, mainly at the very high or very low quantiles. It is clear with this example that increasing the sample size drastically improves the beta estimation.

The intercept beta coefficient can be estimated, but rather than with the other feature beta's, these are not equal over the quantiles. This is because with the data generation method used, there is an inherent random error, namely a normally distributed error with $\mu = 2$ and $\sigma = 2$. This means that it is expected that the intercept beta estimates follow the CDF of a normal distribution. In figure 5.1a, the estimates are shown. These seem to follow the CDF of the described normal distribution, which is an indication that the method works.

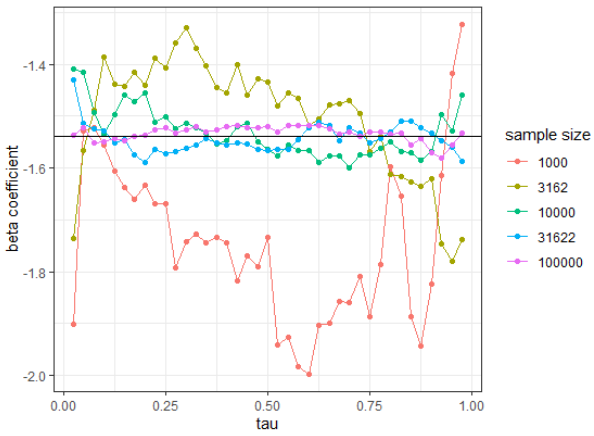
To give a more general overview of the relation between the error in beta estimation and the sample size, the MSE is plotted for all features combined. In figure 5.2 this relationship is shown. As in the previous example, the sample size of 1000 shows bad results, mainly in the outer quantiles, 0.025 and 0.975, however the performance increases quickly and flattens out after 31,622 observations.

Moment generation through quantile regression

With the quantile regression models calculated, it is tested to see if it is possible to reformulate the original distribution. As described in chapter 3, it is possible to translate these β's into mean beta's and variance beta's. As each feature has different beta values for each quantile, these are averaged to give a beta estimation for the mean. This estimation should be close to the feature beta used to generate the dataset.



(a) Intercept beta coefficient estimation



(b) Beta estimation for a single feature

Figure 5.1: Beta coefficient estimation for the intercept and for one variable

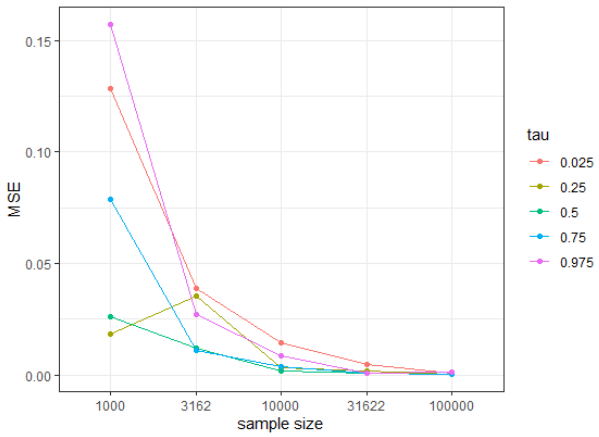


Figure 5.2: Mean square error of beta estimation for multiple quantiles

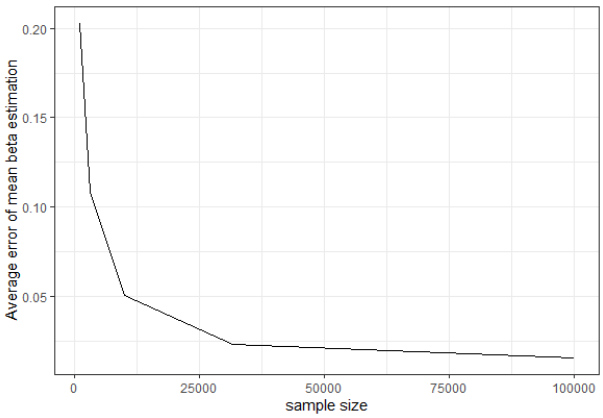
In figure 5.3a the relation between the mean beta estimation and the sample size is shown with on the x-axis the sample size and on the y-axis the average error in estimation over 10 features. The results conform with the hypothesis that quantile regression is able to find the statistical measures mean and variance.

Using these measures, it is possible to calculate the errors as if the beta's estimated are used in a OLS model. This gives two opportunities: calculate the original standard deviation of the error and calculate the R^2. The results are shown in table 5.1. What is also shown in this table is the optimal R^2. This value is calculated by using the input beta's to obtain the true errors. The estimated R^2's are very close to the optimal values and the standard deviation of the errors are very close to the value 2, which was used to generate the data.

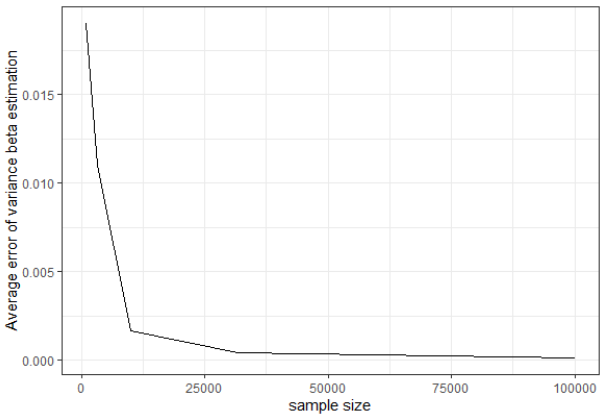
Quantile estimation

The next testing to be done with relation to the number of observations is to see the impact on estimating the quantiles. Using this, a comparison can be made between the two values for each observation. If the quantile regression models are not sufficiently well tuned, the quantile estimates will be deviate more from the true quantiles. To give an impression the quantiles are plotted in figure 5.4a and 5.4b for two different sample sizes: 10000 and 100000. A red line depicts the line $y = x$, the case where the estimate is equal to the actual quantile.

The graphs indicate an improvement when increasing the number of observations. There are three points to notice: The scedasticity is not equal over the quantiles, and at both ends the estimates cut off at 2.5% and 97.5%. The second point is because these are the outermost quantiles estimated through quantile regression. The point about the scedasticity is more difficult. It appears that the quantile regression approach is better at estimating quantiles at the outer ends, than in the middle region. The root mean squared error is taking at 39 sequential intervals to show the change over quantiles. This is plotted in figure 5.5. This is likely due to the fact that quantile estimates have less room for error at the outer quantiles, since the error can only be one way. The third point is that due to the interpolation, there will always be some distortion.

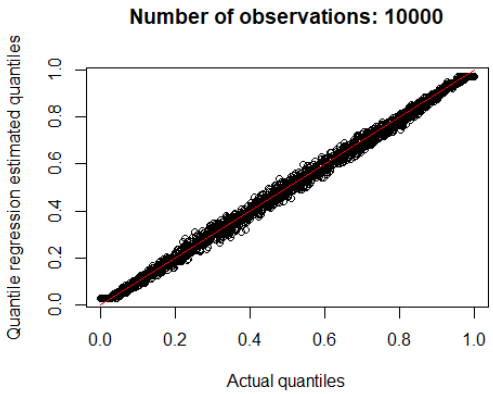


(a) Mean beta estimation

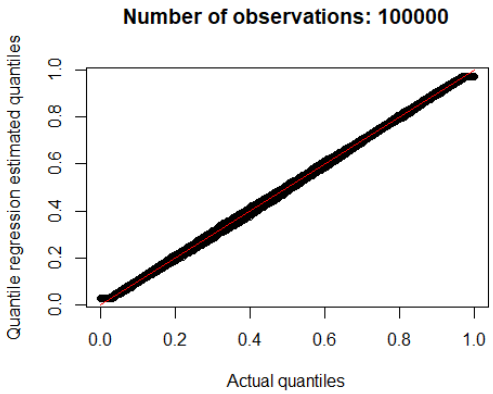


(b) Variance beta estimation

Figure 5.3: The relation between sample size and beta estimation for mean and variance



(a) 10000 observations



(b) 100000 observations

Figure 5.4: Quantile estimates versus theoretical estimates

Calculation times

In the fifth column of table 5.1 the calculation time of quantile regression is shown for different sample sizes. This is visualized in a graph in figure 5.6. The calculation time is of course heavily dependent on the system the script is run. However, the relationship between these two factors is of importance. It seems that the calculation times increase more than linearly. This is important to note, since the models used in this example are all quite simple and there has not been any cross validation implemented.

5.2 Features

The initial results from testing quantile regression show that the method can be used to estimate the quantiles of a quite simple dataset. The next step is to see how well it performs when the number of features is increased and the complexity of the features is increased as well.

In order to see the effect of these parameters, new datasets will be created with the following characteristics: The number of observations will be set at 10,000. The previous testing showed that this sample size gives reasonable results for the simple dataset used. The number of features will be change between the following sets:

- Boolean 10, 20, 50, 100 features
- Numeric 10, 20, 50, 100 features
- An equal mix of both types 10, 20, 50, 100 features

An important part of regression is the ability to ignore features that do not contribute to the model. For this reason, a test will be conducted with 4 added features that have a beta of 0. This means that the feature is

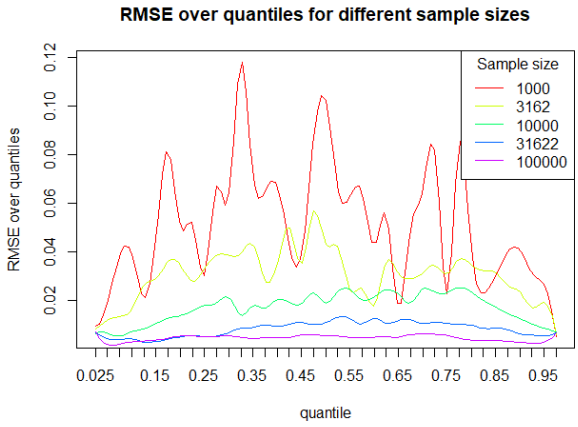


Figure 5.5: RMSE over quantiles for different sample sizes

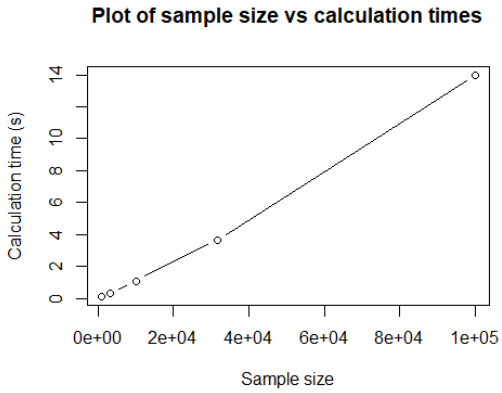


Figure 5.6: Plot of sample size vs calculation times

Table 5.1: The impact of sample size on important performance metrics

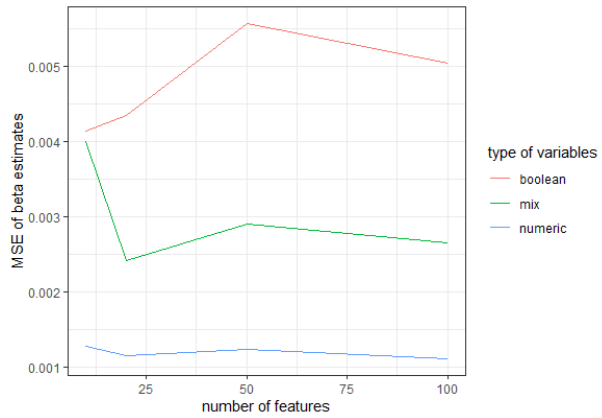
Sample size	Error standard deviation	Estimated R^2	Optimal R^2	Calculation time (s)
1,000	2.084	0.657	0.664	0.08
3,162	2.032	0.670	0.671	0.30
10,000	1.991	0.674	0.675	0.97
31,622	1.997	0.674	0.674	3.61
100,000	1.999	0.675	0.675	14.29

irrelevant for the dependent variable. The goal is to check whether quantile regression accurately finds a beta value of 0 for this feature.

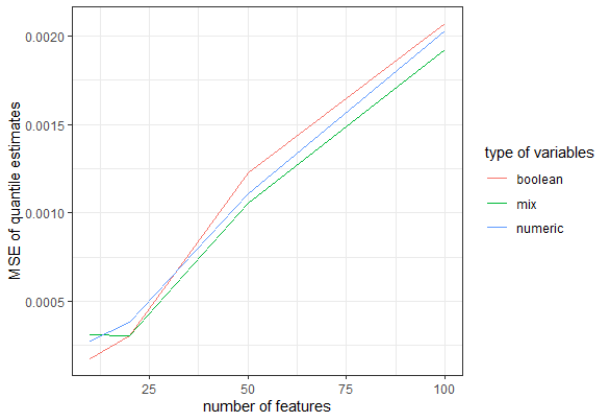
Impact of features on beta estimation

The results of these three tests are shown in figure 5.7a. The dataset with Boolean features is apparently significantly more difficult to model using quantile regression than the numeric dataset. The mixed dataset is of course in the middle. The Boolean dataset shows a drop in error as the number of features is increased, however the total error in quantile estimation is increased due to the number of features increasing as well. Quantile regression was however able to give the irrelevant features a beta value of 0 for all datasets.

It is important to note that while the MSE per feature stays relatively the same, the total MSE of quantile estimation does increase. This is because there are more features that have errors, and these errors are combined into the errors of the quantile estimates.



(a) Number of features vs MSE of beta estimates



(b) Number of features vs MSE of quantile estimates

Figure 5.7: The relation between features and model performance

Impact of features on the calculation times

As was found out in the previous section, quantile regression is computationally heavy for relatively simple models. Increasing the number of features increases calculation times as well. The results are shown in figure 5.8. There does not seem to be a significant difference between the type of features, however the number of feature does significantly affect the calculation time.

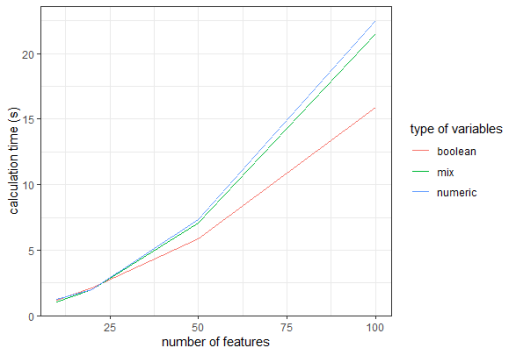


Figure 5.8: The impact of the number of features on the calculation time

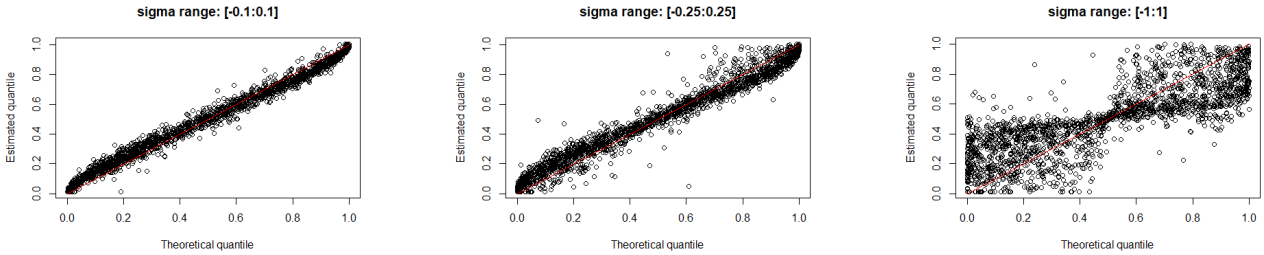
5.3 Type of dependent variable generation method

As mentioned before, there are different ways the values for the dependent variable can be generated. In the previous tests, only the simple linear model was used. In this section, a different method is used as well: a linear model with log-linear variance. This model is generated the same way as the linear model, however each feature gets an additional beta for the variance as well.

Quantile estimation

When applying the same techniques used in the previous sections, the quantiles that are obtained from quantile regression are compared to the ones that are used in the generation process. The other parameters of the dataset are: 10,000 observations, 20 Boolean, and 20 numeric features.

The MSE of the beta's of the mean is 0.00346, which is similar to the result on the linear model. However, depending on the sigma beta's the results vary for quantile estimation varies from good to very bad. Since the variance is log-linear, the effects of the sigma beta's changes rapidly as the value becomes higher. The results of quantile estimation can be seen in the following graphs:



(a) beta sigma range small (b) beta sigma range medium (c) beta sigma range large

Figure 5.9: The relation between features and model performance

These figures differ much, because as mentioned in the previous chapter, quantile regression tries to fit a linear model on an exponential quantile distribution. The closer the exponential component is to 0, the more linearly the quantile behaves. Comparing the quantile estimations on LMVAR distributed data in figure 5.9 to the quantile estimations on the homoskedastic dataset of figure 5.4 the spread is a lot broader. The effect is most prominent at the higher quantiles. in figure 5.9c this is visible. Here quantile regression is still able to determine whether a claim is above or below the median, however that is about it. In figure 5.9b, the effect is less noticeable, however in the quantiles below the median there is a bias to attribute higher quantiles and in the higher quantile a bias to attribute lower quantiles. In practice the more common beta sigma range is the medium version (-0.25 to 0.25).

In conclusion quantile regression performs better on the linear dataset. The reason for this is that the conditional quantiles are not linear in the LMVAR dataset, which causes errors when this effect is too prominent. However, if the sigma's responsible for the conditional variance are low enough, the quantile regression can still find reasonable results.

Chapter 6

Preliminary analysis of available data

Now that the model that will be applied is analyzed, quantile regression will be used on a real dataset. Firstly, an in-depth analysis of the dataset will be given in this chapter.

As mentioned before, Posthuma Partners does statistical analysis of insurance claims. Most of the types of claims that they handle are car insurance. This is also the case in the dataset that will be analyzed in this research. The dataset shows how the claim initially entered the system and how it was ultimately handled. These two are not always the same, since before the claim is paid out, it might be investigated. If there is data missing about the damage, this can later be added. There are three subsets of the data: input, reparation, and main. The data has been gathered from September 2020 till August 2021. In this time more than 35000 claims were handled via the CMF. The company uses the CMF to guide them in choosing which claims to investigate further by giving the company a list of the claims that are, according to the models, in most need of verification. It does this by giving each claim a rating of 1, 2, or 3, which correspond to the green, yellow and red categories as discussed in chapter 1.

6.1 Datasets

The most important dataset is the main dataset. This is an aggregate of all important data about the claims. There are two versions of this set: First and Last. The distinction here is how the claim was initially entered into the system and how it was finally handled. There are two identifiers for the claims: Case number and Log ID. The case number is constant for the claim during the entire process, while the Log ID is given to the claim when it is entered into the system. Whenever the claim is re-entered, due to an update to the claim, it gets a new Log ID. In order to compare the claim from the First and Last subset, the case number will be used. The only claims that are investigated in this research are those that are in both datasets. This leaves a total of 37,453 claims with 45 columns. Of these 45 columns, there are 25 columns generated by internal models that predict the costs, which leaves 20 columns with claim data. Since the Last dataset contains the most complete claims, this dataset will be used to train the models. Ultimately, the models will be tested on the First dataset as well, to see which models will flag which claims.

There are two other datasets available, each with a First and a Last version, just like the main dataset. These are the input dataset and the reparation dataset. These datasets contain more information about the claims. Each claim consists of multiple actions and repairs done to the vehicle. These separate events have their own row in these datasets. The input dataset is about mutation to the claim. These occur when a detail of the claim needs to be changed. This can be price related or something different. The reparation dataset is about which repair methods were used for the claim and how many times. The datasets are linked by the column 'First Log ID'. This is the Log ID the claim received when it was first entered into the system, so the Log ID in the dataset main First.

6.2 Features

The datasets have much information, not all of which is relevant to this research. In this section, a short list of the features that will be used further is discussed. First, the features that relate directly to the vehicle in question are discussed. Then the cause of damage and the associated repairs. Thirdly, the insurance related features are discussed, and finally, the final costs are discussed and compared to the results of the current model. Based on these results combined with the rule based system discussed in the previous chapter, the CMF produces an advice which will also be discussed.

6.2.1 Car related features

The car related features are the brand and model of the car, the fuel type, the type of car, and some numerical values related to the age of the car. In this subsection, these will be further explained.

Car brand

The brand of a car is used as a feature. Originally, there are 72 car brands in the main dataset. However, this can be brought back a bit by removing capital letters and correcting spelling mistakes. This leaves just 54 car brands. In order to reduce complexity of the dataset, the brands that occur fewer than 100 times in the dataset are replaced by the word "Other". This is the case for 425 claims. After this, 27 categories remain, including "Other". The most common car brand is Volkswagen with 4561 claims, equal to 12.9% of all claims. The least frequent car brand is Alfa Romeo, with just 111 claims, or around 0.3% of all claims.

Fuel type

Different cars can have different fuel types. In the original dataset, there are 10 types. Two of which are "#EMPTY" and "Space". The claims with these categories are removed. There are three fuel types that occur fewer times than 100 and are thus replaced by "Other" as well. This leaves 6 levels of the feature fuel type denoted by the codes: B, D, E, J, Z, and Other.

Object type

The object type is a numeric code that describes the type of vehicle. The most common ones are 1: a passenger car, and 14: a van. In the original dataset there are 10 types, however the vast majority is the passenger car and the van. The other categories are replaced by "Other", since they occur fewer than 100 times. Figure 6.1 shows the boxplot of different types of vehicle. It is clear that the code 1 has the most outliers to higher claim amounts, while the "other" category is more spread out with fewer outliers.

Mileage and age

How old a car is in years and how many kilometers a car has driven are two features as well. Both are numeric, however since there are many cars with an age of zero, there is an extra feature added named Age0. This is "TRUE" if the age of the car is 0 years, and "FALSE" if it is not.

The range for mileage is 999,999 kilometers. There are 4 cases where the mileage is exactly this number. There are multiple possible explanations, among which that the program does not allow for larger numbers. Car age has a range of 68 years. However, the number of cars older than 20 years is about 0.1% of all claims.

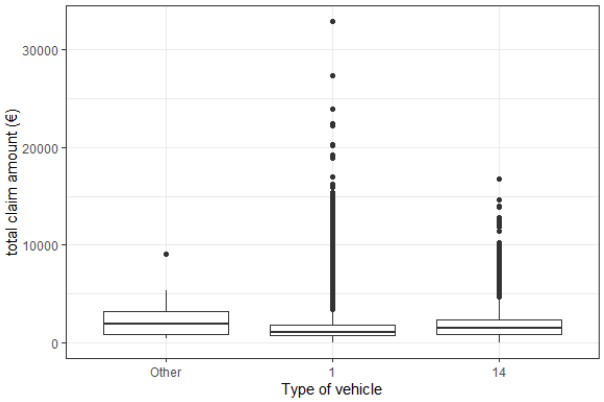


Figure 6.1: Box plot of the relation between type of vehicle and total costs

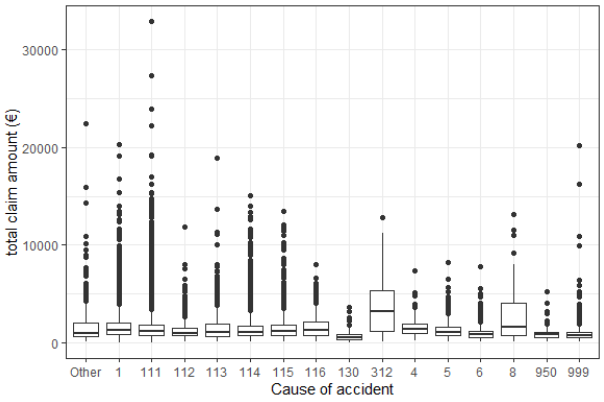


Figure 6.2: Box plot of the relation between cause of accident code and total costs

6.2.2 Accident related features

The next features are accident related. This includes the cause of accident, where the car was damaged (seize point), and the angle of impact.

Cause of accident

The cause of accident is represented by a code. In the initial dataset, there are 31 different codes. One of those is "#EMPTY", which means that it is missing data and thus are those rows removed. There are a lot of codes that are very rare (fewer than 100). These are grouped under "Other". Figure 6.2 shows the box plots of the different causes of the accident. Some have a relatively small range, while others have large outliers.

seize point

The seize point is the point where the vehicle is damaged. In the dataset, this noted by a code which represents that point of damage. In the original dataset, there are 12 codes presented. Except for one, these codes are a string of 5 numbers. The exception is the value "-2000000000". This is supposedly a way of signaling from the person that enters the data to the system that it does not fall in one of the categories. For this reason, claims with this value are removed. One of the codes only occurs 7 times in the entire dataset. This one is replaced by "Other" as well. This leaves 11 levels for this categorical feature.

Angle of impact

There are 14 different angles of impact codes in the dataset. One of those is -2000000000, meaning missing data. These 290 rows are removed from the dataset. The other levels of this feature are reasonably evenly divided. The most infrequent (code 20013) occurs 468 times and the most frequent (code 20012) occurs 6449 times.

Month of claim

Originally, there are two columns describing the timestamp: StartTime and report date. These two provide the same date, which is converted into the Month column. The original columns are removed. The month the claim is entered into the system is used as a predictor of claimed costs. It is conceivable that in the winter, there are different types of car damages than during the summer. Using the months as a predictor can resolve some issues around quantile distribution during the year. If it is true that the claim distribution changes throughout the year, the quantiles will be distributed more evenly through the year by taking the timing into account as well.

6.2.3 Repair related features

There are five repair related features. These features give information about the type of repair that was needed, how long this took, and how intensive the repairs were. Finally, a rating for the repair shop is included as well.

Repair shop

In the original dataset, there are two columns describing the repair shop: Repair shop code and repair shop name. The code column is removed. The repair shop name is used as a feature. In the original dataset there are 1078 repair shops. one of which is the code "#NOTUSED#". This is a code that means the information is missing. There are 148 of such claims, which are removed.

type of repair

Not every car damage is the same, and thus not every method of repair is the same as well. In this dataset, the types of repairs done to the vehicle are represented with a two-letter code. In six cases, it is represented by the code "#EMPTY". These claims are filtered out of the dataset. In total, there are 12 different types of repair codes. However, most claims fall under just two category codes. The combined frequency of the others is fewer than 100. The others are replaced by "Other".

Repair time

The repair time is how long the repair takes in days. This is a numeric value between 0 and 46. There are 5643 cases of the repair taking 0 days. It is assumed that repairs that take 0 days, are different from the others and thus there is a feature added which is TRUE if the repair takes 0 days and FALSE otherwise.

Figure 6.3 shows that the frequency drops quickly after 5 days, where the combined frequency is just 820. After 10 days there are just 51 instances. The relation between repair time and costs was investigated by firstly create dummy variables for each day. This beta coefficients implied a linear relation between the cost and number of days.

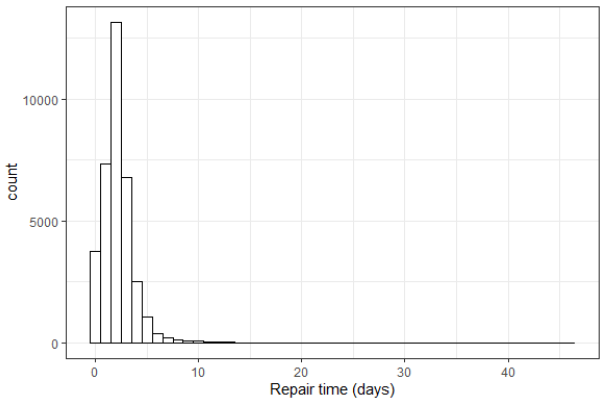


Figure 6.3: histogram of repair time

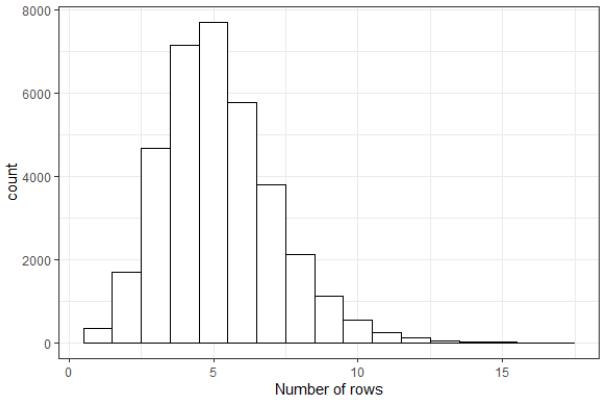


Figure 6.4: histogram of number of rows

Number of rows

As discussed in section 6.1 'Datasets', a claim can have more rows if the repair was complex or if it was mutated many times. The range is between 1 and 17 rows. The minimum is 1, since there is always at least one repair needed. The most frequent number of rows that occurs is 5, after that the frequency falls slowly. Figure 6.4 shows the histogram of the number of rows per claim. This relation is similarly to the repair time assumed to be linearly.

6.2.4 Insurance related features

As mentioned before, the company from which this dataset is, provides their expertise as a service to multiple insurance companies. The next three features are relevant to the insurance for which the company provides its service.

Insurer

In total, there are 42 different insurers in the original dataset represented by numeric codes. There are two large customers with 11022 and 18170 claims, 17 medium-sized customers with between 104 and 1137 claims, and 23 customers with less than 100 claims. The last group is grouped under "Other".

Segment

The insurance claims can be divided into multiple segments, based on the type of insurance. In the original dataset, there are 17, with 12 segments occurring at least 100 times. The less frequently occurring segments are represented by the category "Other".

6.2.5 Costs

The most important aspect of an insurance claim is the claimed cost. In the dataset, the total claimed cost is in the feature 'Total'. The costs are further divided into 'Labour wages', 'Paint costs', 'parts cost'. These three subcategories do not always sum up to the total costs, meaning there are more subcategories. However, those are not available data. The subcategories of the costs are not directly used to monitor the claims, rather they are used as support when further investigation is necessary.

The total claim costs are distributed along a seemingly log-normal distribution. This is graphed in figure 6.5. There are a couple of notable spikes in the distribution, the most significant at 115 euro and 80 euro which occurred respectively 148 and 85 times. The claims with these prices are all from claims claimed by the same car repair shop with very similar details. It is suspected that these claims follow certain price agreements. For this reason claims that have a total claim amount that occurs at least 15 times, is disregarded. Claims that have a predetermined price should not be included in this statistical model. Those claims should be analyzed with the rule-based approach of the CMF. In total there are 299 such claims removed.

6.2.6 Remaining dataset

The dataset that remains consists of 35388 claims and 17 independent variables and 4 dependent variables (costs).

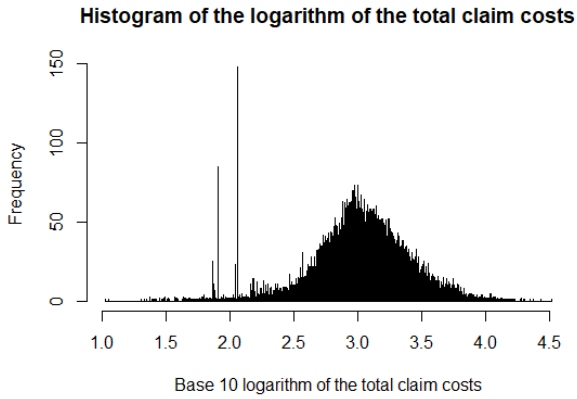


Figure 6.5: Histogram of the logarithm of the total claim costs

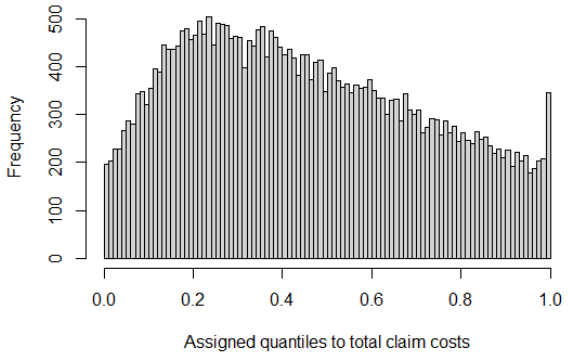


Figure 6.6: histogram of the assigned quantiles of the total cost of claims

6.3 Current model performance

Based on previous years, an internal model was created using LMVAR to calculate quantiles for the claim amounts using data from 2019. The mean estimates and quantile estimates are given for all the claims. These quantiles are calculated for all the separate cost (sub)categories. The results are disappointing, since it performs poorly in two important measures: Quantile distribution and R^2. The quantiles assigned to the total claim amounts are shown in figure 1.4. As mentioned before, the quantiles should, by definition, be distributed uniformly between 0 and 1. Applying the KS-test discussed in 3.2 to compare the conditional quantile distribution with a uniform distribution, the p-value is less than $2.2 * 10^{-16}$. This means that the null-hypothesis of uniformity can reasonably be discarded. As an example, if the company would investigate claims based on the 70th percentile, it would investigate just 21% of the claims.

The other parameter is the R^2. This R^2 is calculated using the formula 1.6.1. This results in an R^2 of 0.04 for the untransformed values and -0.18 for the logarithm of the costs. This value is skewed much due to the corona pandemic and the fact that it was trained on a separate dataset. Historically, the models achieve an R^2 of around 0.49. This means that the model both not very good at distributing quantiles and bad at reducing the variance of the claim costs. It is therefore important to design a new model for the costs.

Chapter 7

Model Design for real data

Now that an overview is given of the real dataset and how the model performs, it is time to describe the model design. The goal is to compare three types of regression models: OLS-regression, LMVAR, and Quantile regression. All models will be trained on a subset of the dataset consisting of 80% of the claims and tested on the remainder.

7.1 Regression models design

In this section, the creation of the regression models is explained. Firstly, OLS regression, then LMVAR, and finally the quantile regression method. For each model the used packages are discussed and the methods to improve the models are mentioned as well. All three models use initially the same dataset, which is explained in the previous chapter.

7.1.1 OLS regression

The OLS regression model is created using the standard 'lm' function in R. The model attempts to predict the conditional expected total claim amount using all the features discussed. Based on this model the Beursch-Pagan test will be applied [5]. This test is meant to measure the null hypothesis of homoskedasticity by creating a linear model on the standardized squared errors of the model. Koenker has created a robust variant in 1981 [6] which is a linear moodel on just the square of the errors. This method works better in non-gaussian errors. If the conclusion of this method is that the linear model is heteroskedastic, that means that it is possible that either LMVAR or quantile regression might work better.

Based on the predictions for the expected claim amount, μ, and the assumption that the error is normally distributed with a constant variance, σ^2, the quantiles, q, can be calculated using the quantile function in formula 7.1. $\phi^{-1}(q)$ refers to the standard normal distribution, with $\mu = 0$ and $\sigma = 1$.

$$F^{-1}(q) = \mu + \sigma\phi^{-1}(q) \tag{7.1}$$

In order to reduce overfitting potential, both LASSO and stepAIC will be used and the differences will be compared. The LASSO method will be done using the glmnet package in R. Firstly, using 10-fold cross validation the optimal λ will be chosen, with which the LASSO regression will be done.

7.1.2 LMVAR

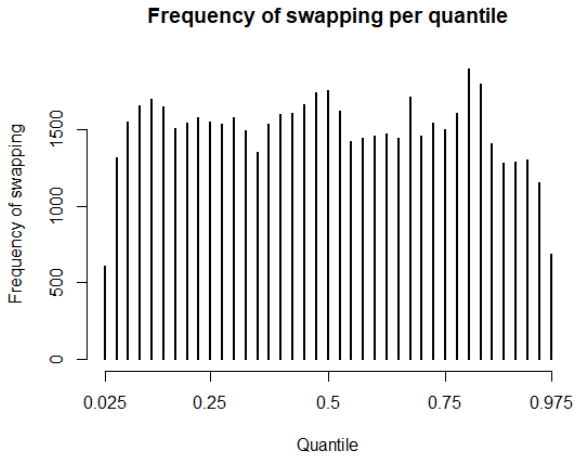
LMVAR uses model matrices as input for both μ estimation and for σ estimation as shown in 1.4.1. The dataframe that was established in the previous chapter is transformed into a model matrix and inputed in the LMVAR model using the LMVAR package in R. Using the models beta's for the mean prediction can be created for each claim. Adding the assumption of normality of errors and the beta's for σ, the conditional quantile, q, can be calculated for each claim as well using the formula 7.1.

In order to reduce overfitting potential, the LMVAR package has included a forward-backward feature selection method. This method attempts to reduce a goodness-of-fit value by alternately removing and inserting features. The goodness-of-fit value used is the AIC[10]. The features are selected for both the μ and the σ.

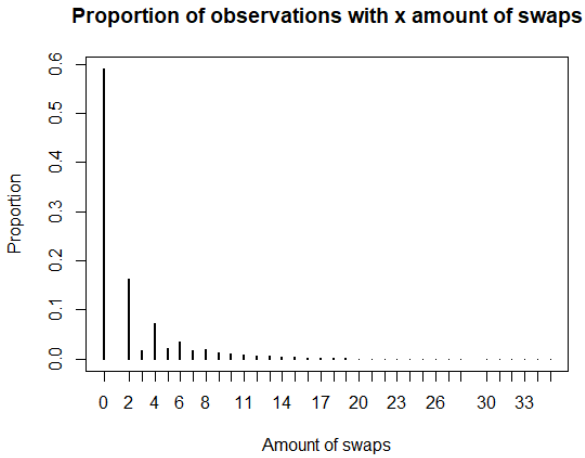
7.1.3 Quantile regression

The quantile regression model was discussed in chapter 4 and 5. Again, the dataset as described in the previous chapter is used as input for quantile regression. The package used is quantreg in R. However, unlike the

previous two models, instead of creating one model for the mean, there are 99 models creating each regression to a different sequential quantile $(0.01, 0.02, \ldots, 0.99)$. This creates a beta estimate for each of the quantile for each feature: a $(p+1) \times \tau$-matrix, where $p+1$ is the number of features (including the intercept) and τ is the number of quantiles modelled. Each model has its own prediction for each claim. Due to the fact that all the models are linear and non-parallel, it occurs that predictions are not monotonously increasing. To solve this, the predictions are sorted, swapping the values between models. In total, there are 11,598 cases in the training set of where the conditional quantile conjugate predictions are not monotonically increasing. This is around 41% of the cases. Figure 7.1a shows that each quantile, except 0.01 and 0.99, are swapped roughly the same amount. These two outermost quantiles of course can only swap one way, which about halves the swaps. Figure 7.1b shows that around 59% of the observations do not require swaps, 16% of observations two predictions are swapped and around 25% require more than two swaps. Naturally, no observation has one swap. In order to calculate the assigned quantiles, the true value is linearly interpolated between the closest two predictions.



(a) Frequency of swapping per quantile



(b) Proportion of observations with x amount of swaps

Figure 7.1: The swaps that occurred when sorting the conditional quantile conjugate predictions

As is shown in 3.1, the beta's from each model can be averaged per feature to calculate the "mean" beta. Using these mean beta's, a mean prediction can be made, similarly to the OLS or LMVAR models. Taking the variance of the beta's per feature gives an indication of the variance that is caused by each feature.

Of course, quantile regression can suffer from overfitting as well. In the previous chapter 2 the option for feature selection was discussed. For each quantile a model is fitted on, the LASSO method will be applied. Using these feature selected models, the previous steps will be repeated.

7.2 Performance comparison methods

Based on the use case of the model, a number of performance comparisons will be designed. In chapter 3 a number of useful tools were described to compare the performance of different models in this case. In this section, the separate comparisons will be described one-by-one.

7.2.1 Feature influence comparison

There are two influences a feature can have: on the expected value and on the skedasticity of the prediction. The OLS-regression model assumes homoskedasticity, meaning the effect of the features is 0 on the variance. The LMVAR assumes log-linear variance. Using the quantile regression approach, the variance effects can be estimated. These effects are assumed to be linear.

The goal of this section is to compare which features are deemed influential for which models and in what way. The three models will also be compared to their 'feature-selected' models.

7.2.2 R^2 with variants

In total, there are 7 models, each with their own R^2. However, because the R^2 rewards overfitting, two alternative R^2s are inspected as well: adjusted R^2 (R^2_{adj}) and out-of-sample R^2 (R^2_{oos}). As a recap, the definition of the R^2 can be found in 1.6.1. The formula for the R^2_{adj} can be found in 7.2.1. N is the number of observations used

for the model and p is the number of features used. The R^2_{adj} is always lower than the regular R^2, since the fraction $\frac{(N-1)}{N-p-1}$ is at most 1. The R^2_{adj} punishes the model for using features that do not improve the model enough.

Definition 7.2.1. R^2_{adj}

- $R^2_{adj} = 1 - (1 - R^2)\frac{(N-1)}{N-p-1}$

In order to see how the model perform on out-of-sample data, or the test set, the R^2_{oos} is used. The R^2_{oos} is calculated like formula 7.2.2

Definition 7.2.2. (R^2_{oos})

- Let SS_{res} be the sum of squares of the residuals of the model on the test set
- Let SS_{tot} be the sum of squares of the difference of values to the sample mean of the train set
- $R^2 = 1$ - $\frac{SS_{res}/n_{test}}{SS_{tot}/n_{train}}$

The goal of comparing the results of the models on these measures is to see how well the models decrease uncertainty.

7.2.3 Quantile distribution

Consistently assigning quantiles to claims uniformly is very important, as discussed in the introduction. To see how well the quantiles are assigned on a test set, the KS test will be used to the range from 0 to 1. The assigned quantiles will be compared to a uniform distribution and if the p-value is below 0.05 the null-hypothesis of uniformity is discarded. The test statistic of the KS test is based on the maximum difference between the empirical CDF and the test CDF. Since the quantile regression method is based on discrete quantile estimation, the quantiles in the tail are rounded towards the closest quantile. This will likely create large bumps at the end of both sides of the histogram. It is, however, for this use case not relevant whether the claim falls in the 99.9'th percentile or the 99.8'th. For this reason, the KS-test will also be applied to the claims in the range of 0.01 to 0.99. This is the range between quantiles that were estimated.

7.2.4 Flag rates and Kappa

The previous two performance measures were mostly aimed at describing the performance of the models on the "Last" dataset. In other words, how the claims are at the end of the process. However, claims enter the system differently in some cases. It is interesting to see which claims are flagged in the "First" dataset. Since the models are trained on a training subset of the "Last" dataset, the testing on the "First" dataset will be done on those claims that are in the test subset. In chapter 3 the flag rates and Kappa were explored. At which quantile the claims are flagged, is dependent on the preference for each insurance company. However, for this testing, all claims will be flagged at 60% and 80%. Since the outcome of the models are now discrete values, the Kappa's can be calculated to see how much the different models agree.

Chapter 8

Results on real data

In the previous chapter, the outline of the modelling on real data was laid out. The regression models are all set up and in this chapter the results are analysed.

8.1 Model results

In this section, the estimated impact of the features is analysed. There are three types of models: OLS regression, LMVAR, and quantile regression. For each, there is a regular approach and a feature selection approach.

8.1.1 OLS regression

A simple regression model was used on a training subset of the dataset in order to make prediction for the total costs. The training set was 80% of the observations of the dataset. The residual standard error is 0.5498 with 28097 degrees of freedom. The log-likelihood of the model is -23127.84. Since this model assumes homoskedasticity, this residual standard error is assumed to be constant. In order to test this assumption, the Beursch-Pagan test is applied. This resulted in a p-value of less than $2.2 * 10^{-16}$, which allows the 0-hypotheses of homoskedasticity to reasonably be discarded. This means that it is likely that this type of regression is not optimal for this dataset.

For this model, a LASSO version was designed as well, to reduce overfitting potential. This was done using the glmnet-package [12]. Firstly, an optimal λ was chosen via cross-validation. The results are shown in figure 8.1. On the x-axis the logarithm of the λ is presented and on the y-axis the resulting MSE of the model. The numbers above the graph represent the number of non-zero beta coefficients.

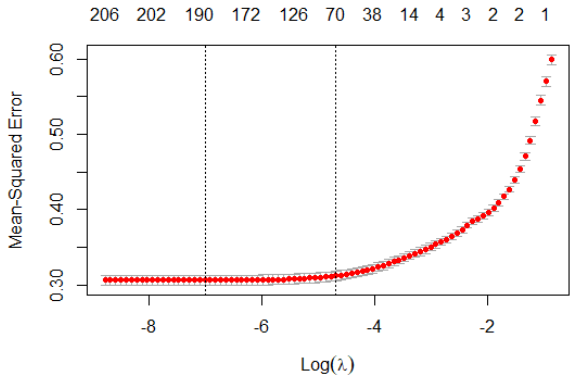


Figure 8.1: CV plot of the lambda's

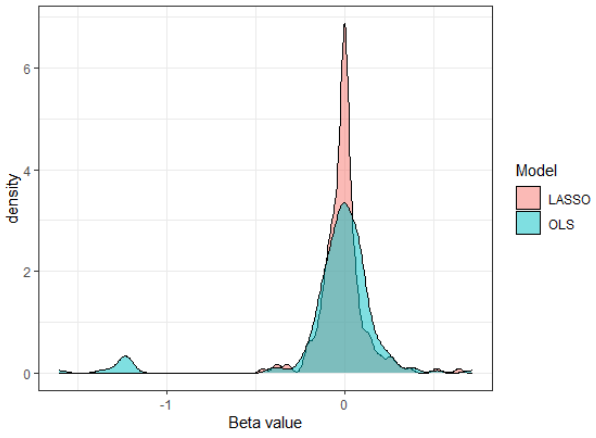


Figure 8.2: OLS LASSO beta values

Features and feature selection

As described in chapter 7, there are two versions of the OLS regression model: regular and LASSO. Figure 8.2 shows the density plots of the beta values for the features, except for the intercept, of both the regular OLS regression model in blue and the LASSO version in red. The most notable difference is the small surface of OLS regression at around -1.2. The features of these β's are the seize points. Where the model without a

L1 regularization deems this aspect to be quite important, their effect is greatly reduced by LASSO. Only one seize point remained a magnitude of above 0.1, with 0.32. This seizepoint also has the largest magnitude in the regular model with 1.62. In total there are 26 features whose beta coefficients were completely reduced to 0. Among these were mostly seizepoints, insurance companies and repairshops. A full list of the beta coefficients per model is included in the appendix.

8.1.2 LMVAR

Similar to the OLS regression, an LMVAR model was trained on the same subset of training data as the previous model. The log-likelihood of the original model is -20138.33, which is higher than that of the OLS regression model. Combined with the degrees of freedom of the LMVAR model, the AIC is 41128.65. In order to decrease this value, the forward-backward stepAIC was applied. This reduced the AIC to 40900.86, by removing 76 beta coefficients for the μ value and 89 beta coefficients for the σ estimation. There are 35 features removed from both the σ and the μ estimation.

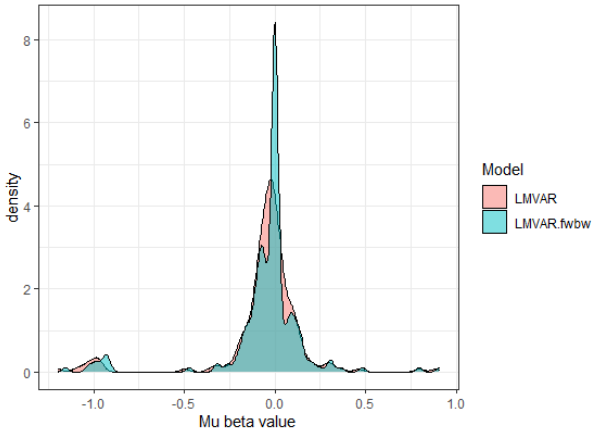


Figure 8.3: LMVAR mu beta values

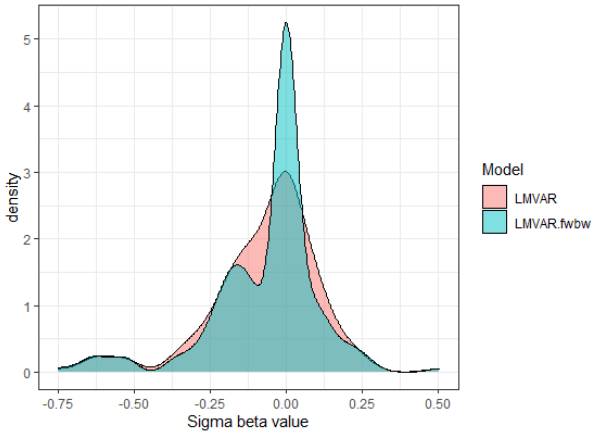


Figure 8.4: LMVAR sigma beta values

Features and feature selection

Figure 8.3 shows the density distribution of the coefficients for the μ beta's of both the regular LMVAR model and of the forward-backward stepAIC model (LMVAR.fwbw). The removed coefficients are given the value 0. As expected, the density is much higher at 0 for the model LMVAR.fwbw than for the other model. Most of the feautres removed were already close to 0, where the largest magnitude is less than 0.1. Just like with the OLS regression model, the features relating to the seizepoint all large negative coefficient. However, unlike the LASSO, the stepIAC function has not reduced the effect of these features to the model.

The density of the coefficients of the β_σ's are shown in figure 8.4. As mentioned in the previous section, there are 89 features removed from the σ estimation. On the left side of the distribution, there is a group of values that have a negative coefficient. These are the 'Cause of accident' categories, ranging from -0.75 to -0.38. These coefficients imply that these value reduce the spread of the total costs the most relative to the category "Other". The list of removed features for the σ estimation includes mostly: Seize points (4), months (6), car brands (18), insurance companies (7), repair shops (46) and accident angle (4).

8.1.3 Quantile regression

The quantile regression model is slightly different from other models. The model consists of a combination of 99 models, each for a different quantile from the set $(0.01, 0.02, ..., 0.98, 0.99)$. Because of this, it is possible for features to affect the total costs differently at different quantiles. The intercept beta coefficients are visualized in figure 8.5. The intercept values do not seem to follow a normal distribution, like in figure 5.1a, for both the original model and for the LASSO model. This implies that the unexplained part of the errors does not follow a normal distribution, meaning that the underlying assumption of normality of OLS regression and LMVAR likely does not hold. The black dotted line is the OLS regression estimate. Figure 8.6 shows the beta coefficients for the feature 'Cause of accident 1'. The original model trends downward more than the LASSO model.

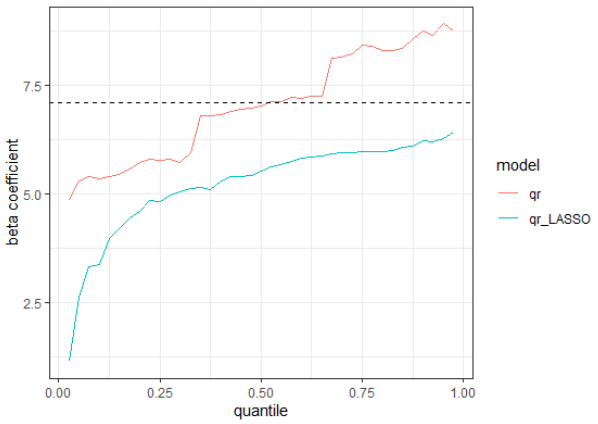


Figure 8.5: Intercept beta coefficient over quantiles

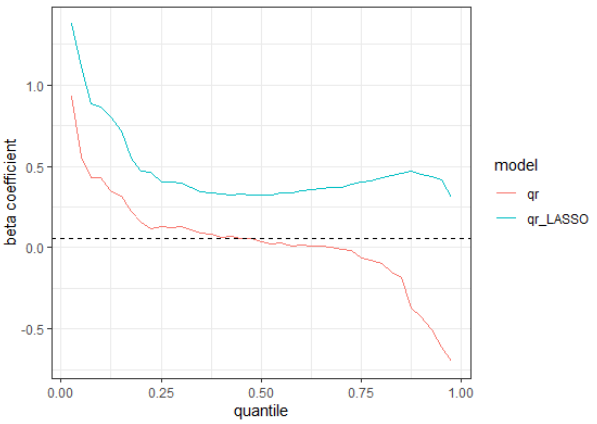


Figure 8.6: 'Cause of accident 1' beta coefficient

In order to be able to compare the results between OLS regression, LMVAR, and quantile regression, the quantile regression β's are translated to μ and β_{σ^2}'s using the methods described in section 3.1. The results are plotted in density graphs in figure 8.7 and 8.8. Just as with the previous models, there is a noticeable area for the μ at around -1.2. These reflect the seize points. Just as with the previous models, the average effect of these values is reduced. However, this is not the case for each quantile.

LASSO has shifted most beta coefficients to the right, which explains why the intercept beta coefficients from figure 5.1a is lower for that model.

The variance within the beta coefficients for a feature, imply an effect on the variance of the predicted outcomes. Figure 8.8 shows the variance for the features. The largest variances are reduced by LASSO, except for three causes (number 8, 312, and 4).

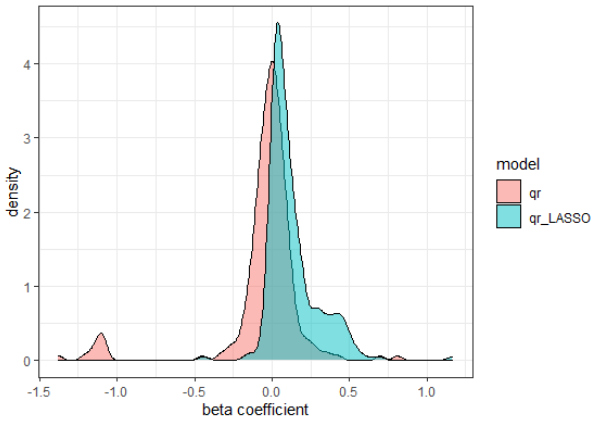


Figure 8.7: Density plot for the μ of the beta coefficients

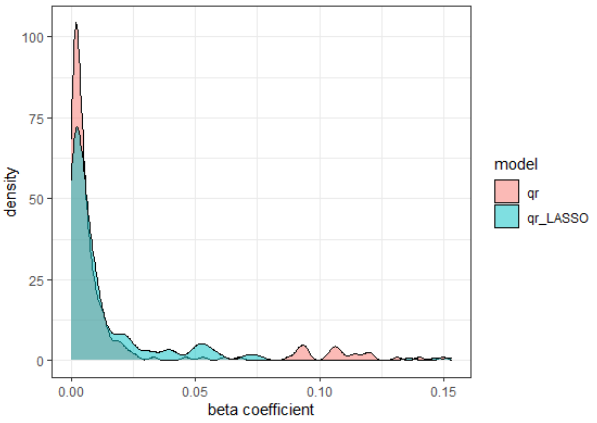


Figure 8.8: Density plot for the variance of the beta coefficients

Since the quantiles conjugate predictions for each claim form a CDF, the PDF can be calculated by taking the slope between each quantile conjugate. This is visualized in figures 8.9 and 8.10. The red line in figure 8.9 is the normal CDF with the mean of the predictions as μ and the standard deviation of the predictions as σ. The predictions follow the normal distribution very closely. This implies that a normal distribution might fit better. Using the density at the correct value, the log-likelihood of this model can be calculated as well by summing the natural logarithm values of all densities. The log-likelihood for this model is -27890. This is significantly lower than that of the LMVAR and OLS regression.

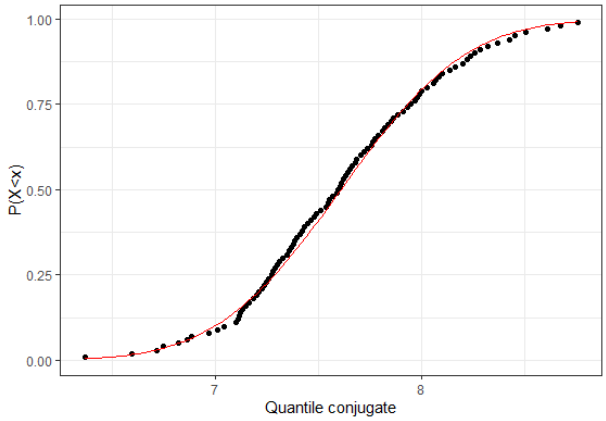


Figure 8.9: Cumulative Distribution Function of one observation

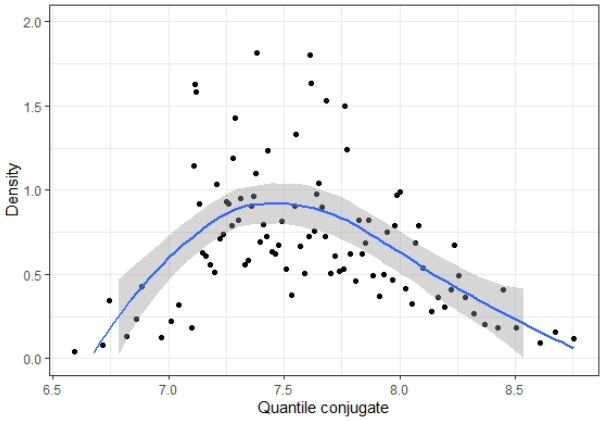


Figure 8.10: Probability Density Function of one observation

8.2 Variance reduction

The goal of regression models is to make predictions of the dependent variable. The true value will often be somewhat different from the prediction. A model that has less variance in this error is better. One key measurement for this is the R^2. As discussed in chapter 7, there will be, on top of the original R^2, two additional types of R^2 measured. These will be calculated for all six previously discussed models (3 regular and 3 with feature selection). For the quantile regression models, the R^2's will be calculated based on the β_μ coefficients.

Table 8.1: Performance of the models measured in R^2

	R^2	R^2_{adj}	R^2_{oos}
OLS regression	0.500	0.496	0.495
OLS with LASSO	0.500	0.496	0.495
LMVAR	0.470	0.453	0.482
LMVAR with fwbw stepAIC	0.469	0.467	0.482
Mean of QR	0.486	0.482	0.491
Mean of QR with LASSO	0.255	0.250	0.232

The R^2's and variants are calculated per model and shown in table 8.1. The results of OLS regression and OLS with LASSO are very close. This is likely due to the fact that the LASSO has only removed 21 features, about 10% of the total amount. These models performed the best on these parameters. Testing on the testing set, the R^2_{oos} is still close to the original R^2 value, so it does not seem to overfit too much.

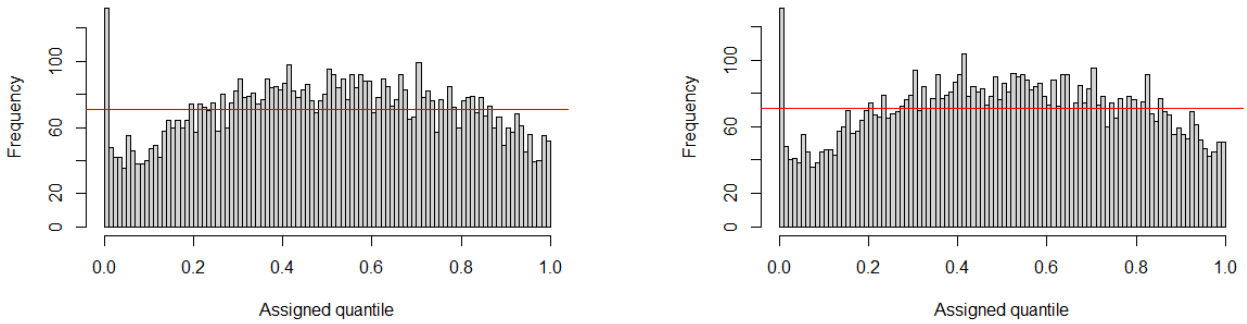
The LMVAR model performs slightly worse, with an R^2 of 0.470 and an R^2_{adj} of 0.453. The adjusted R^2 is calculated with just the μ coefficients. The out-of-sample R^2 is oddly enough slightly higher than the regular R^2. This is due to chance. The LMVAR with fwbw stepAIC, has a similar R^2 of 0.469. Since the feature selection removed 76 μ coefficients, the adjusted R^2 is higher. The out-of-sample R^2 is equal to the original one.

The quantile regression R^2 was the R^2 calculated on the mean predictions of the set of quantile regression models. The R^2 was in between the OLS regression and the LMVAR model. R^2_{adj} and R^2_{oos} show similar results as for the previous models. The LASSO quantile regression models perform very bad. The r squared is just 0.255. The other two variants are also very low. This indicates that the model has not improved, even drastically worsened. It is likely that the methodology of taking the mean of the β coefficients can no longer be applied when LASSO was used. LASSO apparently reduces the continuity between the models.

8.3 Quantile estimation

In this section, the performance of how uniformly the quantiles are distributed between 0 and 1 is discussed per model. The quantiles are assigned to the test set. This means that the model was not trained on these data points. For each model, a histogram is crafted to showcase the frequency of assigned quantiles in the test set. Each histogram has 100 bins, so each bin represents 1 percentile, or 0.01. Every histogram also has a horizontal red line, representing the uniform distribution the assigned quantiles should follow. This is equal to the average of 70.78 observations.

The first models to be inspected are the OLS regression model and the OLS regression with LASSO. By visual inspection of figure 8.11 it becomes clear that both models do not uniformly distribute the quantiles across the observations. The middle part of the quantile range get too many observations. Another noticeable point is that many claims are assigned quantiles of less than 0.01. Apparently, there is a skewness in the claims that is not addressed by these models. The KS-test for the regular OLS regression results is a p-value of 9.326×10^{-15} and for the LASSO variant 1.077×10^{-14}, so for both model the null-hypothesis of uniformity can be discarded.

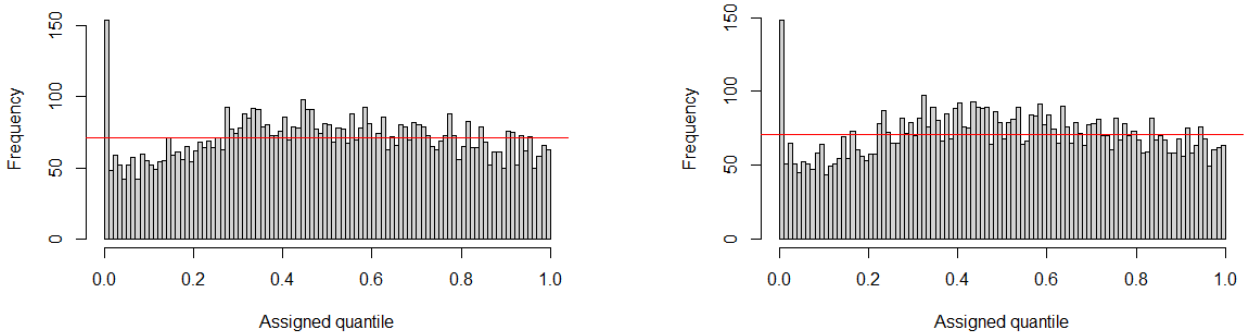


(a) OLS regression

(b) OLS LASSO regression

Figure 8.11: Histograms of the assigned quantiles for the OLS regression models

Next, the LMVAR models are analysed. Figure 8.12 shows that the claims seem to be more uniformly distributed than with the previous model. However, the large quantity of claims getting an assigned quantile of less than 0.01 is still present, which can be seen by the large up tick in the first percentile. The KS-test provide a p-value of 1.721×10^{-08} for the original model and 3.98×10^{-08} for the stepAIC model. These scores are higher than with the previous model, which indicates that the model does perform somewhat better in this aspect. However, the p-values are still much too low, and thus the null-hypothesis for uniformity will be discarded for these models as well.



(a) LMVAR

(b) LMVAR with fwbw stepAIC

Figure 8.12: Histograms of the assigned quantiles for the OLS regression models

The next model to analyse is the quantile regression model. The LASSO variant is discarded, since it performed too badly on the previous parameter: predictive ability. At first glance at figure 8.13 it is clear that the model distributes quantiles much more evenly than the previous models. There is no region of quantiles that seem to have more observations assigned to them, and the large bump at the first percentile is gone as well. This conclusion is underlined by the KS-test which gives for the range between 0 and 1 a p-value of 0.3095, which is means that based on this data, the null-hypothesis of uniformity still stands. Since quantile regression is limited by the number of models that are created, the KS-test will also be done for the range of 0.01 and 0.99, which is between the smallest and largest quantiles that are modelled. This results in a p-value of 0.4162, which is even a larger value.

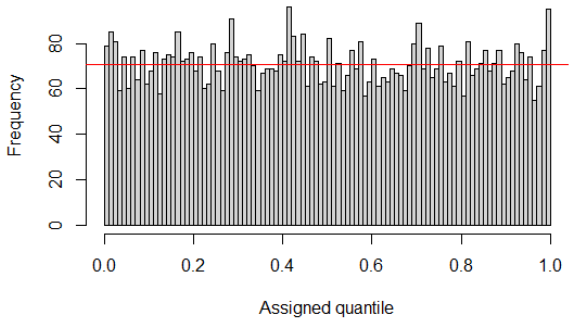


Figure 8.13: Histogram of assigned quantile of quantile regression

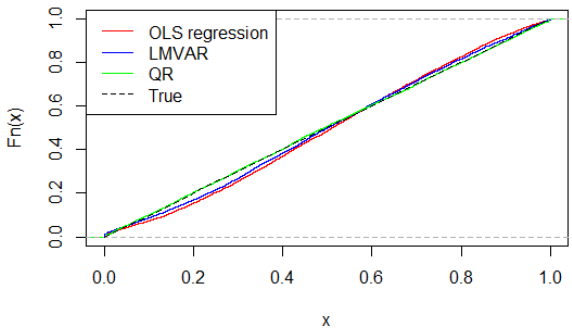


Figure 8.14: Empirical distributions of the different models

A different way of visualizing the previously discussed results is by plotting the empirical CDFs of the models. This is done in figure 8.14. The black dotted line shows the optimal, uniform CDF. This is a straight diagonal line from zero to one. The red and blue lines are the OLS regression and LMVAR models, respectively. On top of the black dotted line, there is the green line representing the quantile regression results, which is clearly the closest imitator of the uniform distribution.

8.4 Flag rates and kappa's

Now that the basic performance of the three models is compared to the final dataset, it is time to look at which claims the models would have flagged when they first entered the system. For this, the dataset "First" is used. This dataset is transformed similarly as the dataset used to train the models. After that, the claims with dossier numbers that are not in the dataset used for training are selected. For this section only the original models are tested: OLS, LMVAR and quantile regression. However, these will also be compared to the average flag rate, which is 15% for a red flag.

The OLS regression model and the LMVAR model have a Cohen kappa of 0.706. OLS with the quantile regression model have a Cohen kappa of 0.441, and finally the kappa between quantile regression and LMVAR is 0.85. These values are to be expected. The low kappa between quantile regression and OLS regression can be explained by the bad quantile distribution of the OLS regression model. The LMVAR model was in between the other two models, so its kappa's is with both other models relatively high.

Both the OLS regression model and LMVAR model flagged too few observations: respectively 12.3% and 13.5%. The quantile regression model flagged 15.4%, which is slightly higher than expected, but close enough. In total, about 9.4% of all claims are flagged by all three models, 4.7% by 2 models and 3.4% by only one model.

The goal of this research was partly to find out if quantile regression could provide a more predictable quantile assignment process. figure 1.2 shows an inconsistent flagging process. Chapter 1 explains how the flags are assigned, but in part this is based on the quantile assigned to the claims. Figure 8.15 shows the different flag rates over time for the different models. The range of flag rates for quantile regression stays between 0.144 and 0.165 (0.21), while the ranges of LMVAR and OLS are larger with respectively 0.125 to 0.15 (0.25) and from 0.11 to 0.135 (0.25).

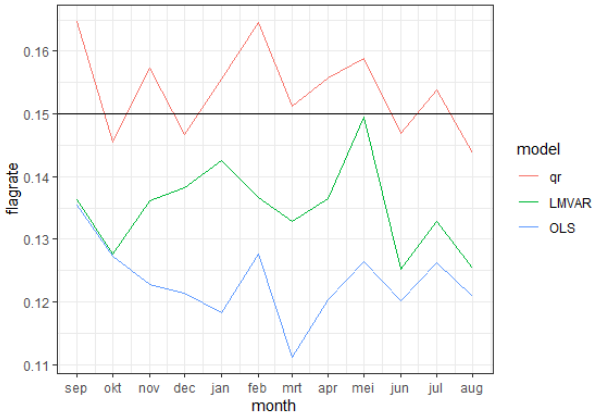


Figure 8.15: Flag rates over time for the three models at quantile 0.85

8.5 Overall results

Overall, each model performed well on different measures. The OLS regression had the best R^2 values, but the worst quantile distribution. The LMVAR had the best log-likelihood, but the worst R^2 values. Finally, the quantile regression models had the best quantile distribution, but the worst log-likelihood. However, the difference in R^2 values was limited. The value for the worst R^2, except for the LASSO quantile regression, was 0.46 and the highest 0.50. The difference in log-likelihood is less important, since the R^2 values are close. The only measure where differences are large is the degree of uniformity among quantiles. Without losing too much predictive ability, the quantile regression model is able to uniformly distribute quantiles. Due to this, the conclusion is that quantile regression performs the best overall.

Chapter 9

Conclusions and recommendations

In this research the possibilities for quantile regression were explored in the context of insurance claim modelling. First the context of the research was investigated. The important qualities of a model for this application were quantified. These are the ability to assign quantiles to insurance claims accurately and uniformly. Next, an understanding of the models workings was established and a framework of performance measurements was established. These measurements were designed to be in line with the use-case. After that the model was tested in a controlled environment where the parameters were known and could be changed to measure the effect on performance. After the results of this testing was in, a real dataset was analyzed and processed. This dataset was used to apply OLS regression, LMVAR, and a set of quantile regression models. By using the performance measurements established in the framework, the performance was compared based on predictive abilities and consistency of quantile assignment.

9.1 Conclusions

From the first research on synthetic data, it became clear that the datasets that are available to Posthuma Partners contain the necessary amount of observations for quantile regression to work. While the sheer amount of modelling to be done, can be draining on computing power, with adequate processors, this can be overcome.

On the real dataset, the three models that were compared performed differently on different measures. The most important aspects the models were measured are: feature importance, predictive ability, and consistent quantile assignment. By using the knowledge gained in the performance measurement chapter, the mean β coefficients and the variance β coefficients were estimated. These allowed quick insight in which features had impact on both the mean total value, but also the skedasticity. The model gave comparable answers to both the OLS regression model as well as the LMVAR model.

While the quantile regression model did not exceed the other models in reducing unexplained variance, the performance was comparable. The model got quite close to the R^2 of the other models, without overfitting on the data, as was concluded based on the R^2_{oos}. The R^2_{adj} for the model based on the means of the quantile β coefficients. It did however, become apparent that the LASSO methodology did not appropriately address the feature selection. The R^2 of the model created with LASSO quantile regression was very low. There needs to be further research into how LASSO can be successfully integrated into this type of methodology.

The final measure of performance was the quantile distribution. Clearly, quantile regression was the best model to do this. The quantiles were very uniformly distributed. This allows for easier and more consistent flagging of claims.

9.2 Recommendations

Overall the recommendation of this research based on the conclusions is to incorporate the methodology of this research around quantile regression into the models of Posthuma Partners. Quantile regression was shown to be close in performance around reducing unexplained variance, good at identifying important features, both for the mean estimation and for the variance, and good at uniformly assigning quantiles to observations. From this research a number of recommendations are proposed. These are split into two parts: Data analysis and quantile regression. The data analysis recommendations consist of ways the data collection and analysis can be improved, while the quantile regression recommendations consists about ways the quantile regression methodology can be applied, fine-tuned, and which other routes can be taken using quantile regression.

Data analysis recommendations

1. It is recommended to be more strict in the way data is imputed into the system. The most clear example is the car brand feature that is used. This feature needed a lot of work to make consistent by removing the effect of capital letters and repairing type errors. By only allowing a predetermined list of car brands, this feature is more usable. It seems that this is already applied to the feature 'car repair shop'.

2. Adding a time-related feature, like the included 'Month' feature can help smooth the quantile distribution over the year. In the current model that was designed in this research, the month feature did not have a large impact on the performance, due to the data only being available for one year. It is therefor necessary that data for multiple years is analyzed to detect seasonal patterns.

3. The final R^2 was not improved by the model. While the number of observations was plenty, the features apparently do not explain the costs well enough. It is recommended that it is investigated which features should be included as well in order to improve the predictive ability of the model.

4. Corona has had a large impact on all aspects of society and through that shown that it is important to keep the models that are used updated. Rather than yearly updated models it is recommended to continuously update the models with new data. With this trends in the data can be captured earlier.

Quantile regression recommendations

1. Applying quantile regression can help reduce fluctuations in workload by more uniformly assigning quantiles. The recommendation is thus that this method can be considered.

2. Testing on the synthetic datasets gave insights in how well the model performed. However, there are more parameters than can be altered. The most important ones are the distribution of the error and the feature creation.

3. Further research needs to be done into feature selection methods. This research did not result in a model with feature selection that gave satisfactory values.

4. An important advancement in quantile regression is the application of additive smoothing. This method is important to look into further.

5. There are many more applications of quantile regressions, such as quantile neural networks, quantile decision trees and random forests with quantile regression.

9.3 Discussion

This research was conducted with the aim of providing an analysis of possible application to the quantile assignment process in screening insurance claims. Firstly, the use case of the model was analyzed. It became clear that the problem could be split into two subproblems: How does quantile regression work in a synthetic environment in which the parameters are known, and how can quantile regression be applied to a real dataset. The problem description phase of the research was a lot about choosing on which aspect to focus. Ultimately the choice was made to keep the research technical and focus on the quantile regression model. However, a different approach that was shortly investigated was to create a financial model to see how different models could reclaim the highest amount of money from incorrect claims. This approach was relatively quickly abandoned, since such a model would be very biased towards the model used in the creation of the dataset. Also from the data that was available about differences between how the claim first entered the system and finally was paid out, there was no relation found between the differences and claim amount and the flag that was assigned.

On the synthetic dataset a lot of testing was conducted which helped in the design of the model on the real dataset and gave an indication of how well the models would perform. However, the number of parameters which were tested upon was limited. An important factor that was not measured is the effect of the unexplained error (ϵ). This was static in the testing and set to have normal distribution with $\mu = 0$ and $\sigma = 2$. The value was chosen such that the combined R^2 of the dataset was similar to real datasets that were available (~ 0.68). For further research it would be interesting to see this effect. Another parameter that was not investigated was the distribution of the features. The Boolean features has a 50/50 chance of being 0 or 1, and the numeric features were distributed according to a standard normal distribution. In real datasets this is not likely. The Boolean features in the real dataset used, were heavily skewed, where the likelihood of a 1 occurring could be as low as 1 in 300. The same holds true for the numeric features. Testing on a dataset that more accurately resembles real datasets is necessary to find out more about the ability of quantile regression to perform on the described performance indicators under different circumstances.

Overall, the design of and testing on the synthetic dataset gave good insight in what the quantile regression model was capable of and whether it was applicable in the real scenario. However, as mentioned before, there are more parameters to be analyzed.

The analysis of the real dataset and the application of quantile regression on which it was applied did not result in an improved performance in the R^2 or log-likelihood. In these two measures, the other models (OLS regression and LMVAR) outperformed the combined quantile regression models slightly. The quantile assignment did improve, the quantiles were much more evenly spread over the range between 0 and 1. This makes the quantile assignment process more predictable and can help reduce operational costs. The flagging rates can see a possible improvement in consistency by using the proposed method.

Bibliography

[1] A. Kolmogorov. "Sulla determinazione empirica di una legge di distributione". In: *Giornale dell' Istituto Italiano degli Attuari* 4 (1933).

[2] Jacob Cohen. "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1 (1960), pp. 37–46. DOI: 10.1177/001316446002000104.

[3] I. Barrodale and F. D. K. Roberts. "Solution of an overdetermined system of equations in the l 1 norm [F4]". In: *Communications of the ACM* 17.6 (1974), pp. 319–320. DOI: 10.1145/355616.361024.

[4] Roger Koenker and Jr. Gilbert Bassett. "Regression Quantiles". In: *Econometrica* 46.1 (1978), pp. 33–50. DOI: https://doi.org/10.2307/1913643.

[5] T. S. Breusch and A. R. Pagan. "A Simple Test for Heteroscedasticity and Random Coefficient Variation". In: *Econometrica* 47.5 (1979), p. 1287. DOI: 10.2307/1911963.

[6] Roger Koenker. "A note on studentizing a test for heteroscedasticity". In: *Journal of Econometrics* 17.1 (1981), pp. 107–112. DOI: 10.1016/0304-4076(81)90062-2.

[7] Stephen Portnoy and Roger Koenker. "The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators". In: *Statistical Science* 12.4 (1997). DOI: 10.1214/ss/1030037960.

[8] Jianqing Fan and Runze Li. "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties". In: *Journal of the American Statistical Association* 96.456 (2001), pp. 1348–1360. DOI: 10.1198/016214501753382273.

[9] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. "Evaluating Kolmogorov's Distribution". In: *Journal of Statistical Software* 8.18 (2003), pp. 1–4. DOI: 10.18637/jss.v008.i18. URL: https://www.jstatsoft.org/index.php/jss/article/view/v008i18.

[10] P. Stoica and Y. Selen. "Model-order selection: a review of information criterion rules". In: *IEEE Signal Processing Magazine* 21.4 (2004), pp. 36–47. DOI: 10.1109/MSP.2004.1311138.

[11] Yichao Wu and Yufeng Liu. "VARIABLE SELECTION IN QUANTILE REGRESSION". In: *Statistica Sinica* Vol. 19,No. 2 (Apr. 2009), pp. 801–817. URL: http://www.jstor.org/stable/10.2307/24308857?refreqid=search-gateway.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. URL: https://www.jstatsoft.org/v33/i01/.

[13] T. Chai and R. R. Draxler. "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature". In: *Geoscientific Model Development* 7.3 (2014), pp. 1247–1250. DOI: 10.5194/gmd-7-1247-2014.

[14] Alexandre Belloni and Victor Chernozhukov. *L1-Penalized Quantile Regression in High-Dimensional Sparse Models*. 2019. arXiv: 0904.2931 [math.ST].

[15] Daniel S. Wilks. *Statistical methods in the atmospheric sciences*. Elsevier, 2019.

[16] Posthuma Partners. URL: https://www.posthuma-partners.nl/. (accessed: 26.04.2021).

[17] Posthuma Partners. URL: https://www.posthuma-partners.nl/claims-management-filter/. (accessed: 26.04.2021).

Appendix A

List of available datasets

Dataset	**Number of observations**	R^2	**Distribution of quantiles**	**Remarks**
Company A Total Paid Amount	20,370	0.56	reasonable	LMVAR with upper bound
Company A Total Labour Costs	20,664	0.06	reasonable	LMVAR
Company A Total Parts Costs	11,812	0.37	reasonable	LMVAR
Company B Total Paid Amount	12,015	0.82	good	LMVAR with upper bound
Company B Market Value	12,477	0.77	good	LMVAR
Company B Total Paid Amount	137	0.91	poor	LMVAR
Company B Windscreen Price	8,277	0.76	reasonable	LMVAR
Company C WS Total Paid Amount	11,400	0.73	good	LMVAR with upper bound
Company C WS Windscreen Price	8,600	0.83	poor	LMVAR with upper bound
Company D Total Costs	30,932	0.86	reasonable	LMVAR
Company D Labour Units	30,769	0.48	reasonable	Gamma with log-link
Company D Paint Cost	28,484	0.65	reasonable	LMVAR
Company D Part Cost	23,534	0.82	poor	LMVAR
Company E Total Costs*	**34,777**	**0.49**	**poor**	**LMVAR (not in production)**
Company E Paint Costs	32,646	0.42	good	LMVAR (not in production)
Company E Parts Costs	25,599	0.49	poor	Gamma with log-link
Company F	18,234	0.26	reasonable	LMVAR with upper bound
Company G Total Costs non-glass	10,378	0.84	poor	LMVAR
Company G Total Costs glass	3,332	0.92	good	LMVAR
Company G Labour Units	12,411	0.65	reasonable / good	LMVAR
Company H Paint Units	9,084	0.61	poor	LMVAR
Company H Costs Paint*	9,149	0.62	poor	LMVAR
Company H Costs Parts	12,344	0.58	reasonable	LMVAR
Company H Total Costs non-glass	62,219	0.50	reasonable	LMVAR
Company H Total Costs glass	24,459	0.43	reasonable	'Combined' model
Company H Labour Units	70,842	0.45	reasonable	LMVAR
Company H Paint Units	58,367	0.53	reasonable / poor	LMVAR (not on log) with lower bound
Company H Costs Parts	77,567	0.56	poor	'Combined model'
Company I Total Costs	65,886	0.71	poor	LMVAR
Company I Costs Parts	67,122	0.74	poor	LMVAR
Company I Costs Paint	59,765	0.57	poor	LMVAR
Company I Labour Units	64,522	0.50	reasonable	LMVAR
Company I Paint Units	59,080	0.55	poor	LMVAR

Appendix B

List of features

Feature	Description	Values
vehicle brand	The name of the vehicle brand	27 levels
Age0	Is the vehicle age 0 years	Boolean
Age	The age of the vehicle in years	0-68 years
Repair time	Repair time in days	0-46 days
Repair time0	Is repair time 0	Boolean
Month	In which month was the case entered into the system	12 levels
Seizepoint	Point at which the vehicle got damaged	11 levels
Fuel type	Which fuel type the vehicle uses	6 levels
Object type	What kind of vehicle was damaged	3 levels
Cause of accident	A numeric code to categorize the damage cause	16 levels
Segment	A numeric code to categorize the segment of the owner of the vehicle	12 levels
Repair shop name	The name of the repair shop that provided the service	93 levels
Type of repair	Alpha-numeric code describing the type of repair conducted	5 levels
Mileage	Number of kilometers the car had driven up to the moment of damage	0-999999 km
Angle of impact	A numeric code describing which angle the car got damaged	13 levels
Insurer	A numeric code describing the insurer	19 levels
Number of rows	The number of rows that are included in the case file	

Appendix C

Code for synthetic datasets

This appendix contains the code that was written in order to do the testing on synthetic datasets, as described in chapter 4.1.

C.1 Creating the synthetic dataset

The first code creates a large synthetic dataset that has plenty of rows and columns, such that the testing can be done without having to create more. The dataset is saved to the file 'df_x' and only has the values for the independent variables. The dependent variables are generated later on.

```
#create a dataframe for X
n_row <- 10^6
df_x <- as.data.frame(matrix(nrow = n_row, ncol = 1))

#boolean features
n_boolean <- 100
boolean_col <- c(1:n_boolean)
df_x[,boolean_col] <- 1

#numeric features
n_num_start <- n_boolean
n_num <- 100
numeric_col <- c(n_num_start : (n_num_start + n_num))
df_x[,numeric_col] <- 1

#name col
colnames(df_x) <- c(paste0(rep("b_", n_boolean),
                           seq(1, n_boolean)),      #boolean variables
                    paste0(rep("num_", n_num),
                           seq(1, n_num))   )       #numeric variables

#fill dataframe with values
set.seed(100); for (i in boolean_col) {df_x[,i] <- rbinom(n_row, 1, 0.5) }

set.seed(100); for (i in numeric_col) {df_x[,i] <- rnorm(n_row, 0, 1) } ; rm(i)

save(df_x, file = "df_x.RData") #save file in order to save time
```

C.2 Quantile regression on subsets of the synthetic dataset

The next R file is the file called 'qr with quantreg.R'. This file first creates dependent variable values for all

```
#packages
library(tidyverse)
library(quantreg)
```

```
#Check performance of quantile regression for
#synthetic datasets with different sample sizes
load("df_x.RData")

ntau <- 39
tau_step <- 1 / (ntau + 1)
tau_seq <- seq(from = tau_step, to = 1 - tau_step, by = tau_step)

col_subset <- c(1:10)
df_x <- as.matrix(df_x[, col_subset])

#set beta's for lm
n_num <- sum(startsWith(colnames(df_x), "b_"))
n_boolean <- sum(startsWith(colnames(df_x), "num_"))
set.seed(100); lm_beta <- runif((n_num + n_boolean), -4, 4)
lm_beta_intercept <- 2
lm_sigma <- 2

#calculate y values
df_y <- matrix(nrow = nrow(df_x)) %>%
  as.data.frame() %>%
  mutate(lm = rnorm(n = nrow(df_x),
                    mean = df_x %*% lm_beta + lm_beta_intercept,
                    sd = lm_sigma)) %>%
  mutate(lm_q = pnorm(lm,
                      mean = df_x %*% lm_beta + lm_beta_intercept,
                      sd = lm_sigma)) %>%
  select(-V1)

#create dataframe
df <- cbind(y = df_y$lm, df_x) %>%
  as.data.frame()

#sample sizes to measure
set_samples <- as.integer(10^seq(3, 5, 0.5))

set_df <-    lapply(set_samples, FUN = function(x)
  sample(nrow(df), size = x))

set_train <- lapply(set_df, FUN = function(x)
  sample(x, size = floor(0.8 * length(x))))

set_test <-  mapply(FUN = function(df_set, train_set)
  df_set[!(df_set %in% train_set)], set_df, set_train)

#quantreg models
qr <- lapply(set_train, FUN = function(train)
  rq(y~., data = df, subset = train,
     method = "fn", tau = tau_seq))

#predict the models on the corresponding test sets
qr_predict <- mapply(FUN = function(model, test_set)
  predict(model,
          newdata = df[test_set, ]),
  qr, set_test)

#sort the predict, such that the quantile conjugates
#are monotonically increasing
qr_predict_sorted <-
  lapply(qr_predict,
         FUN = function(x)
```

```
            as.data.frame(t(apply(t(x), 2, sort)))) %>%
  lapply(FUN = function(x) `colnames<-`(x, tau_seq))

#Linear interpolate the real y values for each qr model
qr_lin_int <- mapply(FUN = function(df_set, test_set, sorted_set)
  sapply(seq_along(test_set), FUN = function(obs)
    approx(x = sorted_set[obs, ],
           xout = df$y[test_set[[obs]]],
           y = tau_seq, ties = "mean",
           yleft = jitter(tau_step / 2),
           yright = jitter(1 - tau_step / 2))$y),
  set_df, set_test, qr_predict_sorted)

################################################################################
#plots

#quantile estimation
plot_i <- 5
plot(x = df_y$lm_q[set_test[[plot_i]]],
     y = qr_lin_int[[plot_i]],
     xlab = "actual quantiles",
     ylab = "assigned quantiles")
segments(x0 = 0, y0 = 0,
         x1 = 1, y1 = 1,
         col = "red")

#plots about beta
beta_df <- lapply(qr, FUN = function(x) as.data.frame(
  cbind(t(coef(x)), `tau` = tau_seq))) %>%
  bind_rows(.id = "sample size") %>%
  mutate(`sample size` = as.numeric(`sample size`)) %>%
  mutate(`sample size` = sapply(`sample size`,
                                FUN = function(x) set_samples[x])) %>%
  mutate(`sample size` = as.factor(`sample size`)) %>%
  `row.names<-`(NULL) %>%
  gather(key = "feature", value = "beta coefficient", -`sample size`, -tau) %>%
  mutate(`true beta coefficient` = #add the true beta coefficients
           rep(c(lm_beta_intercept, lm_beta),
               each = length(set_samples) * length(tau_seq))) %>%
  mutate(`error in beta estimation` = abs(`true beta coefficient` -
           `beta coefficient`))

#intercept beta coefficient estimation over quantiles and sample size
ggplot(subset(beta_df, feature == "(Intercept)"),
       aes(x = tau,
           y = `beta coefficient`,
           colour = `sample size`)) +
  geom_line() + geom_point() +
  geom_abline(intercept = lm_beta_intercept,
              slope = 0) +
  theme_bw()

#plot the beta estimates for one feature over quantiles and sample size
ggplot(subset(beta_df, feature == "b_1"),
       aes(x = tau,
           y = `beta coefficient`,
           colour = `sample size`)) +
  geom_line() + geom_point() +
  geom_abline(intercept = lm_beta[1],
              slope = 0) +
  theme_bw()
```

```
#############################################################
#MSE of beta plot
plotted_quantiles <- tau_seq[c(1, 10, 20, 30, 39)]

beta_df %>%
  subset(feature != "(Intercept)") %>%
  subset(tau %in% plotted_quantiles) %>%
  mutate(tau = as.factor(tau)) %>%
  group_by(`sample size`,
           tau) %>%
  summarise(MSE = mean(`error in beta estimation`^2)) %>%
  ggplot(aes(x = `sample size`,
             y = MSE,
             group = tau,
             color = tau)) +
  geom_line() + geom_point() +
  theme_bw()

rm(plotted_quantiles)

#moving RMSE plots
bins <- seq(0.2, 0.8, 0.2)
moving_measure_df <- mapply(FUN = function(assigned, actual)
  sapply(bins, FUN = function(t)
  sum(
    ((assigned - df_y$lm_q[actual])[which(df_y$lm_q[actual] < t &
                                          df_y$lm_q[actual] > t - min(bins))]
     ) ^ 2) ^ 0.5),
  qr_lin_int, set_test) %>%

  as.data.frame() %>%
  `colnames<-`(set_samples) %>%
  mutate("quantile" = bins) %>%
  gather(key = "sample size", value = "Moving RMSE", -quantile) %>%
  mutate(`sample size` = as.numeric(`sample size`)) %>%
  mutate(`sample size` = as.factor(`sample size`))

moving_measure_df %>%
  ggplot(aes(x = quantile,
             y = `Moving RMSE`,
             colour = `sample size`)) +
  geom_line() + geom_point() +
  theme_bw()

#Error in moment generation through features plots
feature_moments_error <- beta_df %>%
  filter(feature != "(Intercept)") %>%
  group_by(`sample size`) %>%
  summarise(means = mean(`error in beta estimation`),
            vars  = mean(`error in beta estimation`^2) - means^2) %>%
  mutate(`sample size` = as.character(`sample size`)) %>%
  mutate(`sample size` = as.numeric(`sample size`))

ggplot(feature_moments_error, aes(x = `sample size`, y = means)) +
  geom_line() +
  theme_bw() + ylab("Average error of mean beta estimation")
ggplot(feature_moments_error, aes(x = `sample size`, y = vars)) +
  geom_line() +
  theme_bw() + ylab("Average error of variance beta estimation")
```

```
rm(feature_moments_error)

#The error sigma
feature_means <- beta_df %>%
  group_by(feature, `sample size`) %>%
  summarise(mean = mean(`beta coefficient`)) %>%
  mutate(feature = as.factor(feature)) %>%
  mutate(feature = fct_relevel(feature, c("(Intercept)",
                                          "b_1",
                                          "b_2",
                                          "b_3",
                                          "b_4",
                                          "b_5",
                                          "b_6",
                                          "b_7",
                                          "b_8",
                                          "b_9",
                                          "b_10"))) %>%
  arrange(feature) #b_10 was placed incorrectly

sigma_lst <- sapply(set_samples, FUN = function(samples)
  mean((df_y$lm[1:samples] -
    cbind("(Intercept)" = 1, df_x[1:samples, col_subset]) %*%
    feature_means$mean[feature_means$`sample size` == samples]) ^ 2) ^ 0.5
  )

sigma_lst
rm(sigma_lst)

#R squared calculation of the mean estimation
SS_res <- sapply(set_samples, FUN = function(samples)
  sum((df_y$lm[1:samples] -
        cbind("(Intercept)" = 1, df_x[1:samples, col_subset]) %*%
        feature_means$mean[feature_means$`sample size` == samples]) ^ 2))
SS_tot <- sapply(set_samples, FUN = function(samples)
  sum((df_y$lm[1:samples] - mean(df_y$lm[1:samples])) ^ 2))
R_squared <- mapply(function(ss_res, ss_tot) 1 - ss_res / ss_tot,
                    SS_res, SS_tot)


#calculate the theoretical R squared
SS_res.theor <- sapply(set_samples, FUN = function(samples)
  sum((df_y$lm[1:samples] -
         cbind("(Intercept)" = 1, df_x[1:samples, col_subset]) %*%
         c(lm_beta_intercept, lm_beta[col_subset])) ^ 2))
R_squared.theor <- mapply(function(ss_res, ss_tot)
  1 - ss_res / ss_tot,
  SS_res.theor, SS_tot)

R_squared
R_squared.theor

rm(SS_res, SS_res.theor, SS_tot, R_squared, R_squared.theor)

#record calculation times for each train set (~80% of total sample size)
calc_times <- sapply(set_train, FUN = function(train)
  system.time(rq(y~., data = df, subset = train,
       method = "fn", tau = tau_seq)))
calc_times %>%
  `colnames<-`(set_samples) %>%
  data.frame() %>%
```

```
  filter(row.names(calc_times) == "elapsed") %>%
  gather(key = "sample size",
         value = "calculation time")

rm(calc_times)
```

C.3 Quantile regression on different features and number of features

In the next section, the code is shown that created the graphs from section 5.2.

```
#packages
library(tidyverse)
library(quantreg)

#testing on synth data with different features
load("df_x.RData")

ntau <- 39
tau_step <- 1 / (ntau + 1)
tau_seq <- seq(from = tau_step, to = 1 - tau_step, by = tau_step)

col_subsets <- list(c(1:10), c(1:20), c(1:50), c(1:100),
                    c(101:110), c(101:120), c(101:150), c(101:200),
                    c(1:5, 101:105), c(1:10, 101:110),
                    c(1:25, 101:125), c(1:50, 101:150))
sample_size <- 10000

df_x <- as.matrix(df_x[1:sample_size, ])

#set beta's for lm
n_num <- sum(startsWith(colnames(df_x), "b_"))
n_boolean <- sum(startsWith(colnames(df_x), "num_"))
set.seed(10); lm_beta <- runif((n_num + n_boolean), -4, 4)
lm_beta_intercept <- 2
lm_sigma <- 2

#calculate the y's
df_y_m <- lapply(col_subsets, FUN = function(c)
  cbind(1, df_x[, c]) %*% c(lm_beta_intercept, lm_beta[c]))

df_y <- lapply(df_y_m, FUN = function(m)
  rnorm(m, mean = m, sd = lm_sigma))

df_y_q <- mapply(FUN = function(q, m)
  pnorm(q, mean = m, sd = lm_sigma),
  df_y, df_y_m, SIMPLIFY = F)

#create train and test sets
train <- sample(sample_size, floor(0.8 * sample_size))
test <- seq_len(sample_size)[-train]

#create df's
set_df <- mapply(FUN = function(cols, y)
  as.data.frame(cbind(y = y, df_x[, cols])),
  col_subsets, df_y)

#quantreg models
qr <- lapply(set_df, FUN = function(df)
  rq(y~., data = df, subset = train,
     method = "fn", tau = tau_seq))
```

```
#predict the models on the corresponding test sets
qr_predict <- mapply(FUN = function(model, df)
  predict(model,
          newdata = df[test, ]),
  qr, set_df,
  SIMPLIFY = F)

#sort the predict, such that the quantile conjugates
#are monotonically increasing
qr_predict_sorted <-
  lapply(qr_predict,
         FUN = function(x)
           as.matrix(t(apply(t(x), 2, sort)))) %>%
  lapply(FUN = function(x) `colnames<-`(x, tau_seq))

#Linear interpolate the real y values for each qr model
qr_lin_int <- mapply(FUN = function(y_sets, sorted_set)
  lapply(seq_along(test), FUN = function(obs)
    approx(x = sorted_set[obs, ],
           xout = y_sets[[test[obs]]],
           y = tau_seq, ties = "mean",
           yleft = jitter(delta_tau / 2),
           yright = jitter(1 - delta_tau / 2))$y),
  df_y, qr_predict_sorted,
  SIMPLIFY = F)

################################################################################
#plots

#record calculation times for each train set (~80% of total sample size)
calc_times <- sapply(set_df, FUN = function(df)
  system.time(rq(y~., data = df, subset = train,
                 method = "fn", tau = tau_seq)))
calc_times %>%
  data.frame() %>%
  filter(row.names(calc_times) == "elapsed") %>%
  gather(key = "col subset",
         value = "calculation time") %>%
  mutate("number of features" = c(10, 20, 50, 100,
                                  10, 20, 50, 100,
                                  10, 20, 50, 100)) %>%
  mutate("type of variables" = factor(c("boolean", "boolean",
                                        "boolean", "boolean",
                                        "numeric", "numeric",
                                        "numeric", "numeric",
                                        "mix", "mix",
                                        "mix", "mix"))) %>%
  group_by(`type of variables`) %>%
  ggplot(aes(x = `number of features`, y = `calculation time`)) +
  geom_line(aes(color = `type of variables`)) +
  theme_bw() + ylab("calculation time (s)")

rm(calc_times)

#Plot the number of features vs the MSE of the beta estimates
#grouped by type of variables
beta_df <- sapply(qr, coef)
mapply(FUN = function(beta, c)
  mean((beta[-1, ] - lm_beta[c]) ^ 2),
  beta_df, col_subsets) %>%
```

```
  as.data.frame() %>%
  `colnames<-`("MSE of beta estimates") %>%
  mutate("type of variables" = factor(c("boolean", "boolean",
                                        "boolean", "boolean",
                                        "numeric", "numeric",
                                        "numeric", "numeric",
                                        "mix", "mix",
                                        "mix", "mix"))) %>%
  mutate("number of features" = c(10, 20, 50, 100,
                                  10, 20, 50, 100,
                                  10, 20, 50, 100)) %>%
  ggplot(aes(x = `number of features`, y = `MSE of beta estimates`)) +
  geom_line(aes(colour = `type of variables`)) +
  theme_bw()

#Plot the number of features vs the MSE of the quantile estimates
#grouped by type of variables
mapply(FUN = function(as_q, th_q)
  mean((unlist(as_q) - th_q[test]) ^ 2),
  qr_lin_int, df_y_q) %>%
  as.data.frame() %>%
  `colnames<-`("MSE of quantile estimates") %>%
  mutate("type of variables" = factor(c("boolean", "boolean",
                                        "boolean", "boolean",
                                        "numeric", "numeric",
                                        "numeric", "numeric",
                                        "mix", "mix",
                                        "mix", "mix"))) %>%
  mutate("number of features" = c(10, 20, 50, 100,
                                  10, 20, 50, 100,
                                  10, 20, 50, 100)) %>%
  ggplot(aes(x = `number of features`, y = `MSE of quantile estimates`)) +
  geom_line(aes(colour = `type of variables`)) +
  theme_bw()
```

C.4 Quantile regression on a synthetic dataset based on LMVAR

In the next section, the code is shown that created the graphs from section 4.1.1.

```
#packages
library(tidyverse)
library(quantreg)

#Testing different models
load("df_x.RData")

#define features
col_subset <- c(1:20, 101:120)
sample_size <- 10000

df_x_models <- as.matrix(df_x[1:sample_size, col_subset])
rm(df_x)

#betas
mu_intercept <- 2
sigma_intercept <- 0.2 #only used by lmvar

set.seed(100); mu_beta <- runif(length(col_subset), -4, 4)

sigma_range <- c(-0.1, 0.1)
```

```
set.seed(100); sigma_beta <- runif(length(col_subset),
                                   sigma_range[1], sigma_range[2])

#define y values
df_y <- as.data.frame(matrix(nrow = sample_size)) %>%
  mutate(lmvar = rnorm(n = sample_size,
                       mean = df_x_models %*% mu_beta + mu_intercept,
                       sd = exp(df_x_models %*% sigma_beta +
                                  sigma_intercept))) %>%
  mutate(lmvar_q = pnorm(lmvar,
                         mean = df_x_models %*% mu_beta + mu_intercept,
                         sd = exp(df_x_models %*% sigma_beta))) %>%
  select(-V1)

#create train and test set
set.seed(100); train <- sample.int(n = sample_size,
                                   size = floor(.8 * sample_size),
                                   replace = F)
test <- seq_len(sample_size)[-train]

#determine quantile grid
tau <- list()
tau$n <-    39
tau$step <- 1 / (tau$n + 1)
tau$seq <-  seq(from = tau$step,
                to = 1 - tau$step,
                by = tau$step)

#create dataframe for qr and run qr
df <- as.data.frame(cbind(y = df_y$lmvar, df_x_models))
qr <- quantreg::rq(y~., data = df, subset = train,
                           tau = tau$seq, method = "fn")
#make predictions and make monotonically increasing
qr$pred <- predict(qr, newdata = df[test, ])
qr$pred_sorted <- as.matrix(t(apply(t(qr$pred), 2, sort))) %>%
  `dimnames<-`(list(seq_along(test), tau$seq))

#Linearly interpolate true values in the list of predictions
LI_q <- sapply(seq_along(test),
               FUN = function(x) approx(x = qr$pred_sorted[x, ],
                                        xout = df$y[test[x]],
                                        y = tau$seq, ties = "mean",
                                        yleft = jitter(tau$step / 2),
                                        yright = jitter(1 - tau$step / 2))$y)

#make plot
{ plot(x = df_y$lmvar_q[test], y = LI_q,
       main = paste0("sigma range: [",
                     sigma_range[1], ":", sigma_range[2], "]"),
       xlab = "Theoretical quantile",
       ylab = "Estimated quantile")
  segments(x0 = 0, y0 = 0,
           x1 = 1, y1 = 1,
           col = "red")
}
```

Appendix D

Code for synthetic datasets

This appendix contains the code that was written in order to create the dataset that was used in chapter 6 to chapter 8.

D.1 Cleaning the original dataset

In this section, the original datasets were taken and analysed to create a dataframe on which quantile regression was applied.

```
source("Load_data_files.R")

#Libraries
library(tidyverse)

#create a function that changes all factors in a column
#with a lower frequency than the lower limit (100) to "Other"
create_Other <- function(old_list, lower_limit = 100) {
  new_list <- replace(old_list,
                      old_list %in%
                        sort(unique(old_list))[table(old_list) < lower_limit],
                      "Other")
  return(new_list)
}

#pick relevant features
desired_col <- c(4, 5, 7, 9:10, 12, 14, 16, 17, 24,
                 34:37, 40, 19:23, 24:28, 29:32, 45, 3, 1)

df <- Last %>%
  select(all_of(desired_col)) %>%
  filter(dossiernr %in% First$dossiernr) %>%
    #only claims from which the final result is known
  filter(log_id  %in% Invoergegevens_Last$LaatsteLogIDNR) %>%
  filter(!totaal %in%
           as.numeric(levels(as.factor(totaal))[table(totaal) >= 15])) %>%
    #should be continuous
  filter(!expertiseadvies == "0") %>% #should have an expertadvice

  #remove claims with missing data
  filter(!brandstof %in% c("#EMPTY#", "Spatie"))       %>% #Brandstof
  filter(!Aangrijppunt %in% c("-2000000000"))          %>% #aangrijppunt
  filter(!stootrichting %in% c("-2000000000"))         %>% #stootrichting
  filter(!reparateurnaam == "#NOTUSED#")               %>% #reparateurnaam
  filter(!Repduur == "-2000000001")                    %>% #Repduur
  filter(!segment == "40999")                          %>% #segment
  filter(!oorzaak == "#EMPTY#")                        %>% #oorzaak
```

```
  #improve features
  mutate(NaamMerk = toupper(NaamMerk)) %>%
  mutate(NaamMerk = replace(NaamMerk, NaamMerk %in%
                            c("POLSTAR", "POLESTAR_2"), "POLESTAR")) %>%
  mutate(NaamMerk = replace(NaamMerk, NaamMerk %in%
                            c("MERCEDES-BENZ_CARS"), "MERCEDES-BENZ")) %>%
  mutate(NaamMerk = replace(NaamMerk, NaamMerk %in%
                            c("SSANGYOUNG"), "SSANG_YONG")) %>%
  mutate(NaamMerk = replace(NaamMerk, NaamMerk %in%
                            c("MG", "MG."), "M_G")) %>%

  mutate(Merk_Model = paste(NaamMerk, ModelCode, sep = "_")) %>%

  #Leeftijd
  mutate(Leeftijd = replace(Leeftijd, Leeftijd == "2020", "1")) %>%
  mutate(Leeftijd = replace(Leeftijd, Leeftijd == "2021", "0")) %>%

  mutate(Leeftijd0 = Leeftijd == "0")                    %>% #Leeftijd0

  mutate(Month = months(StartTime))                      %>% #Month
  mutate(Repduur0 = Repduur == "0")                      %>% #Repduur0

  ##############################################
  #Place Merk_Model at the correct spot, and remove ModelCode
  select(NaamMerk, Leeftijd0, Leeftijd, Repduur0, Repduur, Month,
         everything(),
         -dossiernr, dossiernr, -Merk_Model, Merk_Model,
         -ModelCode, -log_id)  %>%                    #remove completely

  #put in correct data types
  #correct data types and replace uncommon factors with "Other".
  #Also relevel "Other" as the first level
  mutate(across(everything(), as.character)) %>%
  mutate(across(c(dossiernr, Leeftijd, Repduur,
                  Kilometerstand, AantalRegels, Leeftijd,
                  totaal:percentielonderdelen),
                as.numeric)) %>%
  mutate(across(StartTime, as.Date)) %>%
  mutate(across(where(is.character), create_Other)) %>%
  mutate(across(where(is.character), as.factor)) %>%

  #relevel such that "Other" is the first level
  mutate(across(where(function(x) "Other" %in% levels(x)),
                function(x) relevel(x, ref = "Other"))) %>%
  droplevels()

############################################################

rm(First, Last,
   Invoergegevens_First, Invoergegevens_Last,
   Reparatie_First, Reparatie_Last,
   create_Other, desired_col)
```

D.2 Code for applying quantile regression on the dataset

This code is used to generate the values and plot shown in chapter **??**.

```
source("Model_dataset creation.R")

#libraries
```

```
library(quantreg)
library(moments)

#####################################################################
#Quantile regression
#determine quantile grid
tau <-      list()
tau$n <-    99
tau$step <- 1 / (tau$n + 1)
tau$seq <-  seq(from = tau$step,
                to = 1 - tau$step,
                by = tau$step)

#run rq model
qr <- quantreg::rq(totaal~., tau = tau$seq,
                   data = model_dataset, subset = train,
                   method = "fn")

#make predictions and sort them
qr$pred <- predict(qr, newdata = model_dataset)
qr$pred_sorted <- as.matrix(t(apply(t(qr$pred), 2, sort))) %>%
  `dimnames<-`(list(seq_len(nrow(model_dataset)),
                    tau$seq))

#analyze unsorted predictions
#how many predictions are not monotonically increasing?
sum(apply(qr$pred[train, ], 1, is.unsorted))

#check which quantiles have the most swaps
all_unsorted <- apply(qr$pred[train, ], 1, FUN =
                        function(x) names(which(!(x == sort(x)))))

plot(table(unlist(all_unsorted)),
     main = "Frequency of swapping per quantile",
     ylab = "Frequency of swapping",
     xlab = "Quantile", xaxt = "n")
axis(1, at = c(1,
               ceiling(tau$n / 4),
               ceiling(tau$n / 2),
               ceiling(3 * tau$n / 4),
               tau$n),
     labels = tau$seq[c(1,
                        ceiling(tau$n / 4),
                        ceiling(tau$n / 2),
                        ceiling(3 * tau$n / 4),
                        tau$n)])

length_of_unsorted <- lapply(all_unsorted, length) %>%
  as.data.frame() %>%
  t()

plot(table(length_of_unsorted) / nrow(length_of_unsorted),
     ylab = "Proportion",
     xlab = "Amount of swaps",
     main = "Proportion of observations with x amount of swaps")

rm(all_unsorted,
   length_of_unsorted)

#####################################################################
#estimate quantiles and log-likelihood
```

```
LI_q <- sapply(seq_len(nrow(model_dataset)),
               FUN = function(x) approx(x =    qr$pred_sorted[x, ],
                                        xout = model_dataset$totaal[x],
                                        y =    tau$seq, ties = "mean",
                                        yleft = jitter(tau$step / 2),
                                        yright = 1 - jitter(tau$step / 2))$y)

#histogram of assigned quantiles
hist(LI_q[test], breaks = tau$n,
     main = NULL,
     xlab = paste("Assigned quantile"))
abline(h = length(test) / 100, col = "red")

#to calculate the log likelihood of the qr models
qr$pred_sorted.derivative <- as.data.frame(
  sapply(1 : (ncol(qr$pred_sorted) - 1),
         FUN = function(col)
           sapply(seq_len(nrow(qr$pred_sorted)),
                  FUN = function(row)
                    tau$step /
                    (qr$pred_sorted[row, (col + 1)] -
                       qr$pred_sorted[row, col])))) %>%
  `colnames<-`(tau$seq[-1] - tau$step / 2)

LI_density <- sapply(seq_len(nrow(qr$pred)),
                     FUN = function(x)
                       approx(x = unlist(qr$pred_sorted.derivative[x, ]),
                              xout = LI_q[x],
                              y = tau$seq[-1] - tau$step / 2,
                              rule = 2, ties = "mean")$y)

loglikelihood_of_rqs_model <- sum(log(LI_density[train]))

rm(loglikelihood_of_rqs_model,
   LI_density)

#CDF and PDF plots
obs <- 8

pdf_of_obs <- cbind(x = qr$pred_sorted[obs, -1],
                    y = t(qr$pred_sorted.derivative[obs, ])) %>%
  as.data.frame() %>%
  `colnames<-`(c("Quantile conjugate", "Density")) %>%
  ggplot(aes(x = `Quantile conjugate`, y = Density)) +
  geom_point() +
  geom_smooth() +
  ylim(0, 2) +
  theme_bw()

cdf_of_obs <- as.data.frame(cbind(x = qr$pred_sorted[obs, ], y = tau$seq)) %>%
  `colnames<-`(c("Quantile conjugate", "P(X<x)")) %>%
  ggplot(aes(x = `Quantile conjugate`, y = `P(X<x)`)) +
  geom_point() +
  theme_bw() +
  stat_function(fun = pnorm,
                args = list(mean(qr$pred_sorted[obs, ]),
                            sd(qr$pred_sorted[obs, ])),
                col = "red")
rm(pdf_of_obs,
   cdf_of_obs,
   obs)
```

```
#############################################################################################
#conditional moments
#var from Stats package is sample var, but we need population var
pop_var <- function(x) mean(x ^ 2) - mean(x) ^ 2

cond_moments <- as.data.frame(cbind(coefficients =
                                      rownames(qr$coefficients))) %>%
  mutate(means = apply(qr$coefficients, 1, mean)) %>%
  mutate(vars =  apply(qr$coefficients, 1, pop_var))

#R squared calculation based on the mean estimations
qr$pred_mean <- apply(model.matrix(totaal~., model_dataset), 1,
                      function(x) x %*% cond_moments$means)
qr$pred_mean_resi <- model_dataset$totaal - qr$pred_mean

#R squared
SS_res <- mean((model_dataset$totaal[train] - qr$pred_mean[train]) ^ 2)
SS_res_oos <- mean((model_dataset$totaal[test] -
                      qr$pred_mean[test]) ^ 2)
SS_tot <- var(model_dataset$totaal[train])

qr_R2 <- list()
qr_R2$R2 <- 1 - SS_res / SS_tot
qr_R2$R2_adj <- 1 - (1 - qr_R2$R2) *
  (length(train) - 1) /
  (length(train) - (length(cond_moments$means) - 1) - 1)
qr_R2$R2_oos <- 1 - SS_res_oos / SS_tot

rm(SS_res,
   SS_res_oos,
   SS_tot)

#############################################################################################
#TESTING FEATURE SELECTION METHODS
qr_LASSO <- quantreg::rq(totaal~., data = model_dataset,
                         tau = tau$seq,
                         method = "lasso")

cond_moments.LASSO <- cbind(coefficients =
                              rownames(qr_LASSO$coefficients)) %>%
  as.data.frame() %>%
  mutate(means = apply(qr_LASSO$coefficients, 1, mean)) %>%
  mutate(vars =  apply(qr_LASSO$coefficients, 1, pop_var))


#R squared calculation based on the mean estimations
pred_qr_LASSO_mean <- apply(model.matrix(totaal~., model_dataset), 1,
                            function(x) x %*% cond_moments.LASSO$means)
pred_qr_LASSO_mean.residuals <- model_dataset$totaal - pred_qr_LASSO_mean

#R squared LASSO
SS_res <- mean((pred_qr_LASSO_mean.residuals[train])^2)
SS_res_oos <- mean((model_dataset$totaal[test] -
                      pred_qr_LASSO_mean[test])^2)
SS_tot <- var(model_dataset$totaal[train])

qr_LASSO_R2 <- list()
qr_LASSO_R2$R2 <- 1 - SS_res / SS_tot
qr_LASSO_R2$R2_adj <- 1 - (1 - qr_LASSO_R2$R2) * (length(train) - 1) /
  (length(train) - (length(cond_moments.LASSO$means) - 1) - 1)
```

```
qr_LASSO_R2$R2_oos <- 1 - SS_res_oos / SS_tot

rm(SS_res,
   SS_res_oos,
   SS_tot)

#beta coefficient density plots
cbind(qr = cond_moments$means[-1],
      qr_LASSO = cond_moments.LASSO$means[-1]) %>%
  as.data.frame() %>%
  gather(key = "model", value = "beta coefficient") %>%
  ggplot(aes(`beta coefficient`, fill = model)) +
  geom_density(alpha = 0.5) + theme_bw()

cbind(qr = cond_moments$vars[-1],
      qr_LASSO = cond_moments.LASSO$vars[-1]) %>%
  as.data.frame() %>%
  gather(key = "model", value = "beta coefficient") %>%
  ggplot(aes(`beta coefficient`, fill = model)) +
  geom_density(alpha = 0.5) + theme_bw()

#beta coefficient plot over quantiles
f <- 60

rbind(qr$coefficients[f, ], qr_LASSO$coefficients[f, ]) %>%
  t() %>%
  as.data.frame() %>%
  `colnames<-`(c("qr", "qr_LASSO")) %>%
  mutate(quantile = tau$seq) %>%
  gather(key = "model", value = "beta coefficient", -quantile) %>%
  ggplot(aes(x = quantile, y = `beta coefficient`)) +
  geom_line(aes(color = model)) +
  geom_hline(yintercept = lm$coefficients[f], linetype = "dashed") +
  theme_bw()

rm(f)

#KS testing
ks.test(LI_q, punif)
ks.test(LI_q[which(LI_q > 2 * tau$step & LI_q < 1 - 2 * tau$step)],
        punif, 2 * tau$step, 1 - 2 * tau$step)

#remove values
rm(cond_moments,
   cond_moments.LASSO,
   LI_q, tau,
   pop_var,
   test, train,
   model_dataset, model_dataset.matrix, df,
   pred_qr_LASSO_mean, pred_qr_LASSO_mean.residuals)

#Final plot
{ plot(ecdf(lm$pred$q[test]), col = "red",
       xlim = c(0, 1),
       main = NULL)
  plot(ecdf(LMVAR$pred$q), col = "blue", add = T)
  plot(ecdf(LI_q[test]), col = "green", add = T)
  plot(punif, col = "black", lty = 2, add = T)
  legend(legend = c("OLS regression", "LMVAR", "QR", "True"),
         col = c("red", "blue", "green", "black"),
         lty = c(1, 1, 1, 2),
```

```
        x = "topleft")
}
```

D.3 Code for OLS regression

This code produces the graphs and values for the OLS regression model.

```
#load the same dataset as with the other models
source("Model_dataset creation.R")

#run lm
lm <- lm(totaal~., data = model_dataset, subset = train)
lm$pred <- predict(lm, newdata = model_dataset) %>%
  as.data.frame()                                %>%
  `colnames<-`(c("mu")) %>%
  mutate(q = pnorm(model_dataset$totaal,       #calculate quantiles
                   mean = mu,
                   sd = sigma(lm)))  %>%
  mutate(res = mu - model_dataset$totaal)

library(glmnet)
lm_lasso_cv <- cv.glmnet(y = model_dataset$totaal[train],
                         x = model_dataset.matrix[train, ])
lm_lasso <- glmnet(y = model_dataset$totaal[train],
                   x = model_dataset.matrix[train, ],
                   lambda = lm_lasso_cv$lambda.min)

log(lm_lasso_cv$lambda.min)
plot(lm_lasso_cv)

rm(lm_lasso_cv)

#LASSO predictions with quantiles
lm_lasso$pred <- predict.glmnet(lm_lasso,
                                newx = model_dataset.matrix) %>%
  as.data.frame() %>%
  `colnames<-`(c("mu")) %>%
  mutate(q = pnorm(model_dataset$totaal,
                   mean = mu,
                   sd = sigma(lm_lasso))) %>%
  mutate(res = mu - model_dataset$totaal)

#####################################################################
#coefficient table
lm_coef <- merge(as.data.frame(as.matrix(coef(lm_lasso))),
                 as.data.frame(lm$coefficients),
                 by = 0, all = T) %>%
  replace(is.na(.), 0) %>%
  `colnames<-`(c("Feature", "LASSO", "OLS")) %>%
  mutate("difference" = abs(OLS - LASSO))

#Density plot for the feature beta coefficients
lm_coef %>%
  filter(Feature != "(Intercept)") %>%
  select(LASSO, OLS) %>%
  gather(key = "Model", value = "Beta value") %>%
  ggplot(aes(x = `Beta value`, fill = Model)) +
  geom_density(alpha = 0.5) +
  theme_bw()
```

```
rm(lm_coef)

################################################
#calculate R2
SS_tot <- var(model_dataset$totaal[train])

lm_R2 <- list()
lm_R2$R2 <-  summary(lm)$r.squared
lm_R2$R2_adj <- summary(lm)$adj.r.squared
lm_R2$R2_oos <- 1 - mean(lm$pred$res[test] ^ 2) / SS_tot

#redo for LASSO
lm_lasso_R2 <- list()
lm_lasso_R2$R2 <- lm_lasso$dev.ratio
lm_lasso_R2$R2_adj <- 1 - (1 - lm_lasso_R2$R2) *
  (lm_lasso$nobs - 1) / (lm_lasso$nobs - lm_lasso$df - 1)
lm_lasso_R2$R2_oos <- 1 - mean(lm_lasso$pred$res[test] ^ 2) / SS_tot

rm(SS_tot)

#plot histogram of quantile distribution
hist(lm$pred$q[test], breaks = 100,
     main = NULL,
     xlab = paste("Assigned quantile"))
abline(h = length(test) / 100, col = "red")

hist(lm_lasso_pred$q[test], breaks = 100,
     main = NULL,
     xlab = paste("Assigned quantile"))
abline(h = length(test) / 100, col = "red")

ks.test(lm$pred$q, punif)
ks.test(lm_lasso_pred$q, punif)
```

D.4 Code for LMVAR models

The code produces the graphs and values for the LMVAR models.

```
#load the same dataset as with the other models
source("Model_dataset creation.R")

#run LMVAR model
library(lmvar)
LMVAR <- lmvar::lmvar(y = model_dataset$totaal[train],
                                  X_mu = model_dataset.matrix[train, ],
                                  X_sigma = model_dataset.matrix[train, ],
                                  intercept_mu = T, intercept_sigma = T)

LMVAR$pred <- predict(LMVAR,
                      X_mu = model_dataset.matrix,
                      X_sigma = model_dataset.matrix) %>%
  as.data.frame() %>%
  mutate(q = pnorm(model_dataset$totaal,
                   mu,
                   sigma))  %>%
  mutate(res = model_dataset$totaal - mu)

#feature selection
LMVAR_fwbw <- fwbw(LMVAR, AIC)
```

```
LMVAR_fwbw$pred <- predict(LMVAR_fwbw$object,
                           X_mu    = model_dataset.matrix,
                           X_sigma = model_dataset.matrix) %>%
  as.data.frame() %>%
  mutate(q = pnorm(model_dataset$totaal,
                   mu,
                   sigma))  %>%
  mutate(res = model_dataset$totaal - mu)

#COEFFICIENT PLOTS
#Mu coefficient density plot
LMVAR_coef_mu <- merge(as.data.frame(LMVAR$coefficients_mu),
                       as.data.frame(LMVAR_fwbw$object$coefficients_mu),
                       by = 0, all = T) %>%
  `colnames<-`(c("Feature", "LMVAR", "LMVAR_fwbw")) %>%
  replace_na(list(`LMVAR_fwbw` = 0))

LMVAR_coef_mu %>%
  filter(Feature != "(Intercept)") %>%
  select(LMVAR_fwbw, LMVAR) %>%
  gather(key = "Model", value = "Mu beta value") %>%
  ggplot(aes(x = `Mu beta value`, fill = Model)) +
  geom_density(alpha = 0.5) +
  theme_bw()

rm(LMVAR_coef_mu)

#sigma coefficient density plot
LMVAR_coef_sigma <- merge(as.data.frame(LMVAR$coefficients_sigma),
                          as.data.frame(LMVAR_fwbw$object$coefficients_sigma),
                          by = 0, all = T) %>%
  `colnames<-`(c("Feature", "LMVAR", "LMVAR_fwbw")) %>%
  replace_na(list(`LMVAR_fwbw` = 0))

LMVAR_coef_sigma %>%
  filter(Feature != "(Intercept)") %>%
  select(LMVAR_fwbw, LMVAR) %>%
  gather(key = "Model", value = "Sigma beta value") %>%
  ggplot(aes(x = `Sigma beta value`, fill = Model)) +
  geom_density(alpha = 0.5) +
  theme_bw()

rm(LMVAR_coef_sigma)


#plot histogram of quantiles
hist(LMVAR$pred$q[test], breaks = 100,
     main = NULL,
     xlab = paste("Assigned quantile"))
abline(h = length(test) / 100, col = "red")

hist(LMVAR_fwbw$pred$q[test], breaks = 100,
     main = NULL,
     xlab = paste("Assigned quantile"))
abline(h = length(test) / 100, col = "red")

ks.test(LMVAR$pred$q[test], punif)
ks.test(LMVAR_fwbw$pred$q[test], punif)

#R squareds
r_squared_adj <- function(R2, n, k) {
```

```
  return(1 - (1 - R2) * (n - 1) / (n - k - 1))
}

LMVAR_R2 <- list()
LMVAR_R2$R2 <- pputils::r_squared(LMVAR)
LMVAR_R2$R2_adj <- r_squared_adj(LMVAR_R2$R2,
                                 length(train),
                                 length(LMVAR$coefficients_mu))
LMVAR_R2$R2_oos <- 1 -
  mean(LMVAR$pred$res[test] ^ 2) /
  var(model_dataset$totaal[train])


LMVAR_fwbw_R2 <- list()
LMVAR_fwbw_R2$R2 <- pputils::r_squared(LMVAR_fwbw$object)
LMVAR_fwbw_R2$R2_adj <-
  r_squared_adj(LMVAR_fwbw_R2$R2,
                length(train),
                length(LMVAR_fwbw$object$coefficients_mu) - 1)
LMVAR_fwbw_R2$R2_oos <- 1 -
  mean(LMVAR_fwbw$pred$res[test] ^ 2) /
  var(model_dataset$totaal[train])

rm(r_squared_adj)
```