SFB 649 Discussion Paper 2017-006

RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods

Lukas Borke*

*Humboldt-Universität zu Berlin, Germany

This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 "Economic Risk".

http://sfb649.wiwi.hu-berlin.de
ISSN 1860-5664

SFB 649, Humboldt-Universität zu Berlin
Spandauer Straße 1, D-10178 Berlin

RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods *

Lukas Borke †

Abstract

In order to integrate and facilitate the research, calculation and analysis methods around the Financial Risk Meter (FRM) project, the R package **RiskAnalytics** has been developed. Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and some macro variables. The derived "Risk Analytics" can help to forecast and evaluate the systemic risk for the corresponding markets. The visualization and the up-to-date FRM can be found on `http://frm.wiwi.hu-berlin.de`. Supplementary R codes are published on www.quantlet.de with the keyword FRM. The **RiskAnalytics** package is a convenient tool with the purpose of integrating lasso penalized quantile regression methods with full solution paths and cluster computing support around the topic "Risk Analytics and FRM".

Keywords: Risk Analytics, FRM, Data Analytics, Systemic Risk, Quantile Regression, Lasso, Value at Risk, Parallel and Cluster Computing, EDA, Data Visualization

JEL: C21, C51, G01, G18, G32, G38.

*Financial support from the Deutsche Forschungsgemeinschaft (DFG) via SFB 649 "Ökonomisches Risiko", IRTG 1792 "High-Dimensional Non-Stationary Times Series" and Sim Kee Boon Institute for Financial Economics, Singapore Management University are gratefully acknowledged.

†Research associate at Ladislaus von Bortkiewicz Chair of Statistics, C.A.S.E. - Center for Applied Statistics and Econometrics, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany. Email: lukas@borke.net

1 Software implementation in R

Koenker and Mizera (2014) survey some recent developments of convex optimization and describe some implementations of these methods in R. Quadratic programming (QP), as part of convex optimization, involves the minimization of a positive semi-definite quadratic objective function subject to polyhedral constraints. There are many applications of QP in statistics, typically involving Gaussian likelihoods constrained by some form of linear inequalities. Shape constrained regression examples have gained recent attention, and the introduction of sparse regularization methods like lasso, has greatly stimulated interest in computational methods for such problems. One of the most familiar statistical QP applications in recent times has been the lasso estimator of Tibshirani (1996).

Standard quantile regression (QR) models can be estimated with the `rq()` function of the **quantreg** package (Koenker, 2016). However, software implementations for computing solution paths of lasso penalized QR are rare. **hqreg** (Yi, 2016) is such an example. This R package is relatively new (it was published for the first time on 21 June 2015), its version history is trackable via `https://github.com/cran/hqreg/commits/master`. The main advantage is its C optimization: `https://github.com/cran/hqreg/blob/master/src/hqreg.c`. Yi and Huang (2015) demonstrate both the convergence properties of the proposed algorithm and the numerical experiments, showing that their package implementation is very efficient and scalable to ultra-high dimensions.

Another available R implementation is the supplementary code of Li and Zhu (2008). At the time of the early stage development of the FRM project (Yu et al., 2017), only the latter code was known and available. Therefore, the current lasso penalized QR implementation of FRM relies on the idea of Li and Zhu (2008). In the following, numerical experiments and benchmarks will be provided in order to evaluate the speed and efficiency of the current FRM version.

2 RiskAnalytics package

In order to integrate and facilitate the research, calculation and analysis methods around the FRM project (Yu et al., 2017), the R package **RiskAnalytics** (Borke, 2017) has been developed. Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and the macro variables as described in the Data section in Yu et al. (2017). The derived "Risk Analytics" can help to forecast and evaluate the systemic risk for the corresponding markets.

As member of the Research Data Center (`https://rdc.hu-berlin.de`) Lukas Borke was involved in the development of the FRM project from the very beginning, having the main tasks: automation of data collection, optimization and parallelization of code, and data visualization. Based on this experience, the functionality of the **RiskAnalytics** package is subdivided into 4 major software components:
1) data processing (*get_data.R*);

2) parallel computing (*parallel_calculation.R*);
3) QR methods (*qrL1.R*);
4) "Risk Analytics" (*analytics.R*);

Every software component contains several related functions. Their interaction is presented in Listings 1, 2, 3, 4 and 5.

2.1 RiskAnalytics package: data extraction and analysis part

Listing 1 demonstrates the data extraction and analysis part of the **RiskAnalytics**. The functions `get.nasdaq.companies`, `get.yahoo.data` and `get.macro.data` are responsible for real time processing of NASDAQ, Yahoo Finance and Federal Reserve Bank of St. Louis data. `get.nasdaq.companies` extracts the top NASDAQ companies (sorted by their market capitalization) from the web resource `http://www.nasdaq.com/screening/companies-by-industry.aspx?industry=Finance` by means of the package **RCurl** (Lang and the CRAN team, 2016). `get.yahoo.data` provides daily log returns of the selected NASDAQ companies by use of the package **quantmod** (Ryan, 2016). `get.macro.data`, in its turn, employs both approaches: Yahoo Finance via **quantmod** for the download of the VIX, GSPC (S&P500) and IYR (iShares Dow Jones US Real Estate) macro variables, and direct downloads of the other 3 macro variables from the corresponding web resources on `https://fred.stlouisfed.org/`.

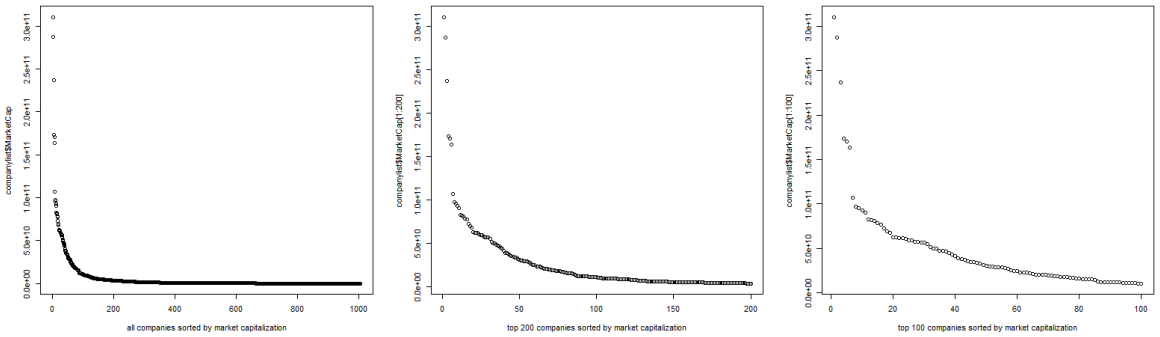

Figure 1: NASDAQ companies sorted by the market capitalization: all (left), top 200 (middle) and top 100 (right), produced via *get.nasdaq.companies*

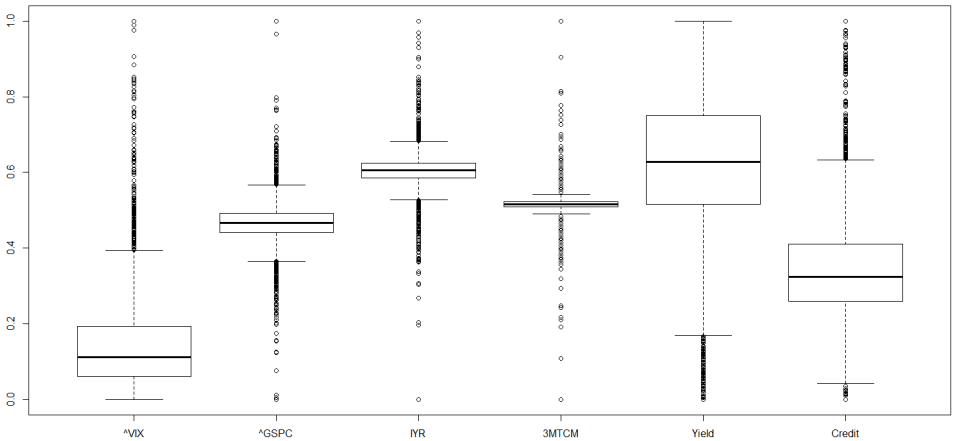

Figure 2: Box plots of macro variables produced via *get.macro.data*

The helping function `combine.data` combines all previously obtained time series in an appropriate time and date format. Additionally, the dimension and the preview of a sub sample of the resulting data frame object is displayed. The latter can be controlled by the parameter `summary_dim`, see also Listing 1. All aforementioned functions provide additional information and, where appropriate, graphical plots for better audit and validation checks of the extracted data, see e.g., Figures 1 and 2.

```
#-----------------------------------------------
# Initialization
#-----------------------------------------------
library(snow)
library(RiskAnalytics)
work_dir = "c:/r/frm/2017"
max_companies = 100

#-----------------------------------------------
# Load data
#-----------------------------------------------
companylist = get.nasdaq.companies()

system.time( yahoo_data <- get.yahoo.data(companylist, max_comp_num = max_companies,
    from_date = "2006-12-29") )
# truncated output for illustration
[1] "97 : SBNY"
[1] "98 : ZION"
[1] "diff length : CIT"
[1] "diff length : APO"
[1] "99 : WRB"
[1] "100 : SEIC"
      User     System    Elapsed
      7.85       1.44      58.20

system.time( macro_data <- get.macro.data(from_date = "2006-12-28") )
      User     System    Elapsed
      1.05       0.06       5.13

final_data = combine.data(yahoo_data, macro_data, summary_dim = c(1:3, 102:107))
[1] "Dimension of the final data: 2534 * 107"
        Date         JPM         WFC       ^VIX     ^GSPC      IYR     3MTCM
1  03/01/2007 0.002290948 0.005049110 0.02353107 0.4414408 0.5978950 0.4904459
2  04/01/2007 0.002493227 0.001677339 0.03029449 0.4577132 0.6204483 0.5159236
3  05/01/2007 -0.008335091 -0.005602276 0.02282655 0.4695943 0.6035441 0.5222930
4  08/01/2007 0.003342404 -0.002812915 0.03170354 0.4337065 0.5632682 0.5286624
5  09/01/2007 -0.004179761 0.002532009 0.02973087 0.4744425 0.6035341 0.4840764

data.analytics(yahoo_data, macro_data)
# truncated output for illustration, correlation matrix of the macro var's
       ^VIX ^GSPC  IYR 3MTCM Yield Credit
^VIX   1.00 -0.14 -0.11 -0.06 0.26 0.55
^GSPC -0.14  1.00  0.81 -0.02 0.00  0.01
IYR   -0.11  0.81  1.00 -0.04 0.01  0.02
3MTCM -0.06 -0.02 -0.04  1.00 0.00  0.00
Yield  0.26  0.00  0.01  0.00 1.00  0.36
Credit 0.55  0.01  0.02  0.00 0.36  1.00
```

Listing 1: RiskAnalytics application example: data extraction and analysis part

The function `data.analytics` from Listing 1, which is actually a part of the "Risk Analytics" software component, provides descriptive statistics for both the NASDAQ companies and the macro variables. All statistical information vital for the subsequent QR methods is summarized in a brief overview. For instance, it becomes immediately obvious that the macro variables will be dominant regressors due to their larger Euclidean norms, compared to those of the NASDAQ companies (see the box plots in Figure 3). Together with the output in Figure 2 and 4, one can easily conclude that the macro variables VIX (1), "Yield spread" (3) and "Credit spread" (4) (see the Data section in Yu et al. (2017) for the enumeration assignment) will be "driving factors" in the QR process because of their high variances. Furthermore, `data.analytics` returns also the correlation matrix of all six macro variables, revealing that the aforementioned variables VIX, "Yield spread" and "Credit spread" have positive correlations among each other, see Listing 1. In the light of this technical analysis, it is hardly surprising that both the FRM and VIX time series reveal a similar behavior, see the "FRM versus VIX" section in Yu et al. (2017).

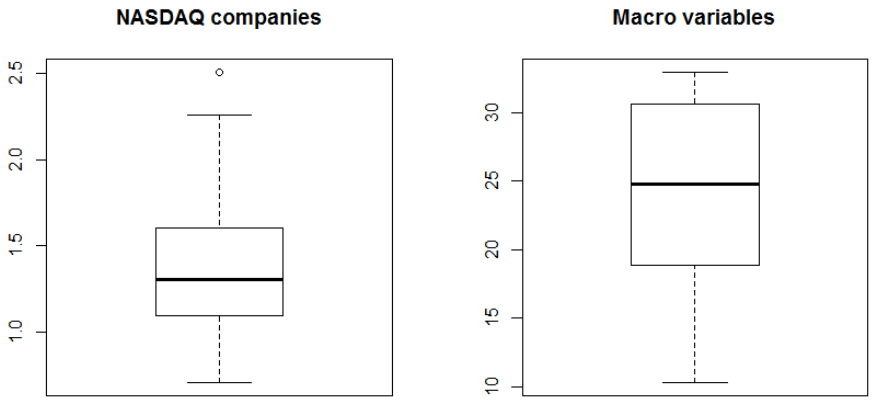

Figure 3: Box plots of the euclidean norms of the Yahoo Finance data/companies (left) and the macro variables (right), produced via *data.analytics*

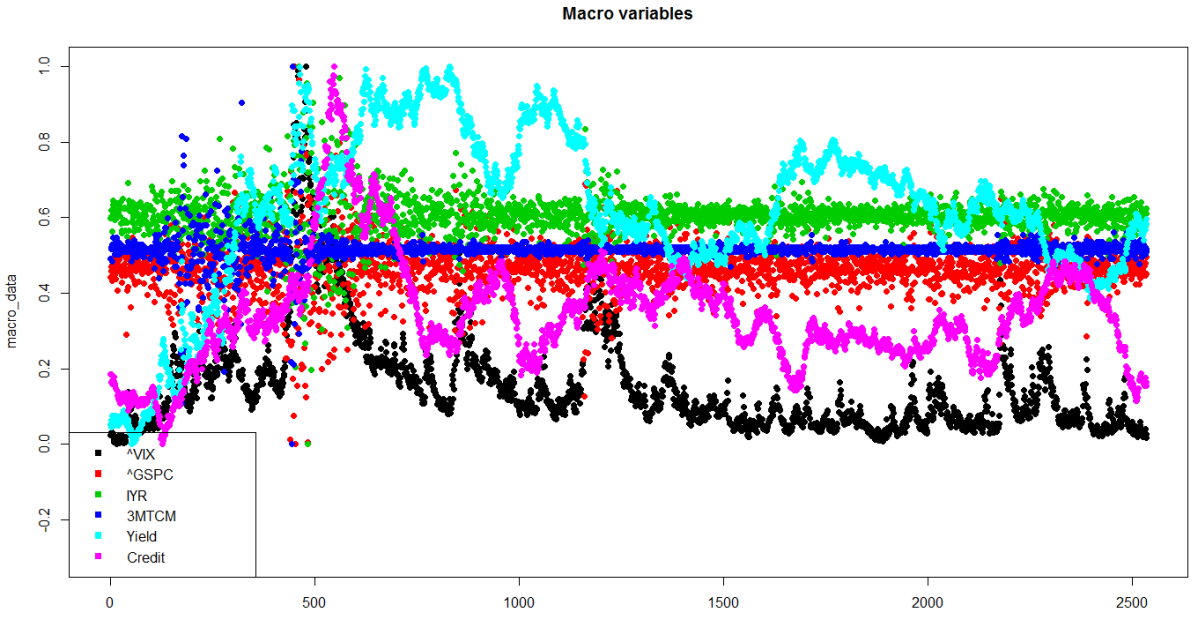

Figure 4: Plot of the macro variables, produced via *data.analytics*

According to Listing 1, the data processing component extracts all needed data in around one minute. Additionally, the `data.analytics` function provides statistical information for the further QR process. All obtained data are stored in the RAM, hence no further write or storage operations are required, and the real time data can be passed over to the next component: parallel computing.

2.2 RiskAnalytics package: parallel computing part

Listing 2 shows the execution and benchmark results of the parallel computing component of **RiskAnalytics**. Based on the packages **snow** (Tierney et al., 2016) and **snowfall** (Knaus, 2015) and the lasso penalized QR implementation of Li and Zhu (2008), the calculation of the QR method is performed for all moving windows and all NASDAQ companies. The most important parameters of the function `parallel.lasso.computation` are `max_companies`, `new_days`, `parallel_cpu`, `p`, `winsize` meaning: 1) number of desired NASDAQ companies, the parallelization is performed along this dimension; 2) number of desired moving windows within the total data observation time frame; 3) number of available CPU's for the parallel computing via **snowfall**; 4) desired quantile value for the QR method; and 5) the length of the moving window. Most of these parameters have default values as displayed in Listing 2.

```
#------------------------------------------------
# Calculate data
#------------------------------------------------

# by default: new_days = 5, parallel_cpu = 4, p = 0.05, winsize = 60

# main calculation for the FRM visualization
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir,
     new_days = 2469, parallel_cpu = 32, winsize = 63)
# R Version: R version 3.3.2 (2016-10-31)
# snowfall 1.84-6.1 initialized (using snow 0.4-2): parallel execution on 32 CPUs.
# Stopping cluster
#    user  system elapsed
#    1.45    3.43 54895.78

# test benchmark for 200 working days, ca. 10 months
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir,
     new_days = 200, parallel_cpu = 32)
# Stopping cluster
#    user  system elapsed
#    0.14    0.14 4287.58

# test benchmark for 5 working days, 1 working week
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir,
     parallel_cpu = 32)
# Stopping cluster
#    user  system elapsed
#    0.17    0.14  115.89
```

Listing 2: RiskAnalytics application example: parallel computing part

For each company and each moving window the QR results are stored in the data structure `parResult`. The latter is basically a list with elements corresponding to the companies.

Every list element j contains the lambda values (λ_j) and beta coefficients (β_j) from the QR procedure. For a given company j, the lambda values are a vector enumerated by the calculated days `new_days`, and the beta coefficients are a matrix, whose rows are the calculated days `new_days` and whose columns are the regressors/covariates.

According to Li and Zhu (2008), the computational complexity of the L_1-norm QR algorithm is $\mathcal{O}(p \min(n,p)^3)$, with n being the length of the moving window and p the number of covariates. The main calculation for the FRM visualization is performed with $n = 63$ and $p = 105$ (99 companies except the regressed one and 6 macro variables), see also Listing 2. In addition to the basic complexity, we have to deal with two further dimensions, i.e. n_c (`max_companies` or *number* of *companies*) and n_w (`new_days` or *number* of moving *windows*).

In summary, the `parallel.lasso.computation` function for the main calculation of the *FRM* lambda time series has a computational complexity of

$$\mathcal{O}(n_c n_w)\mathcal{O}(p \min(n,p)^3), \tag{1}$$

which results in approximately 6.5×10^{12} basic calculations, if we compute the *FRM* lambda for $n_w = 2469$ (around 10 years).

The time complexity benchmarks for four cases: $n_w = 2469$, $n_w = 200$, $n_w = 10$ and $n_w = 5$ are provided in Table 1, see Listing 2 for some examples. The tests were performed on a RDC (Research Data Center, `https://rdc.hu-berlin.de`) Windows server with 16 physical and 32 logical cores and Intel Xeon CPU E5-2690 0 @ 2.90 GHz. In each case `max_companies` was equal to 100, `parallel_cpu` = 32, `p` = 0.05. The corresponding length of the moving window n (`winsize`) and the number of moving windows n_w (`new_days`) are given in the table columns.

n (window size)	n_w	time in seconds	time in minutes	time in hours
60	5	116	2	0.03
60	10	222	4	0.06
60	200	4288	71	1.19
63	2469	54896	915	15.25

Table 1: Time complexity benchmarks for *parallel.lasso.computation* of the **RiskAnalytics** package

As can be expected from Formula 1, the running time of `parallel.lasso.computation` scales in proportion to n_w. For a better comparison, the same time measurements are displayed in seconds, minutes and hours, respectively. As main results of the time complexity benchmarks, we can conclude that:
I) The lasso penalized QR implementation in FRM can be performed within 2 minutes for the calculation of 5 working days and within 15 hours for a time period of 10 years, which shows that the QR calculation is feasible on a contemporary computer with 16 physical cores.
II) For the increase of the speed, only the physical CPU cores are relevant, what means that the calculations can be performed on a usual home PC with 4 CPU cores, like for example Intel Core i5-2500 with 4 physical cores. In this case, the time demand must be multiplied by factor of 4 (16 cores $\div$ 4 cores).

III) The memory demand for the storage of all necessary data and calculation results is very modest and is mainly dictated by the dimensions of the data matrices and frames and the data structure `parResult`. Saved as files, the data object `final_data` from Listing 1 and `parResult` from Listing 2 require around 2 MByte and 60 MByte, respectively.

The `parallel.lasso.computation` function accepts some additional optional parameters for minor validation outputs and allowing to save the calculation results as file outputs. By default, the parallel computing component operates as an "in-memory application" without requiring any disk Input/Output operations.

2.3 RiskAnalytics package: QR.analytics part

The code examples in Listing 3 demonstrate the *QR.analytics* part of **RiskAnalytics**, the former being a subset of the "Risk Analytics" software component. QR.analytics comprises 3 functions: `QR.regressors.stats`, `QR.beta.stats` and `QR.variance.vs.beta`. The output `parResult` from the parallel computing part serves as an "object of investigation".

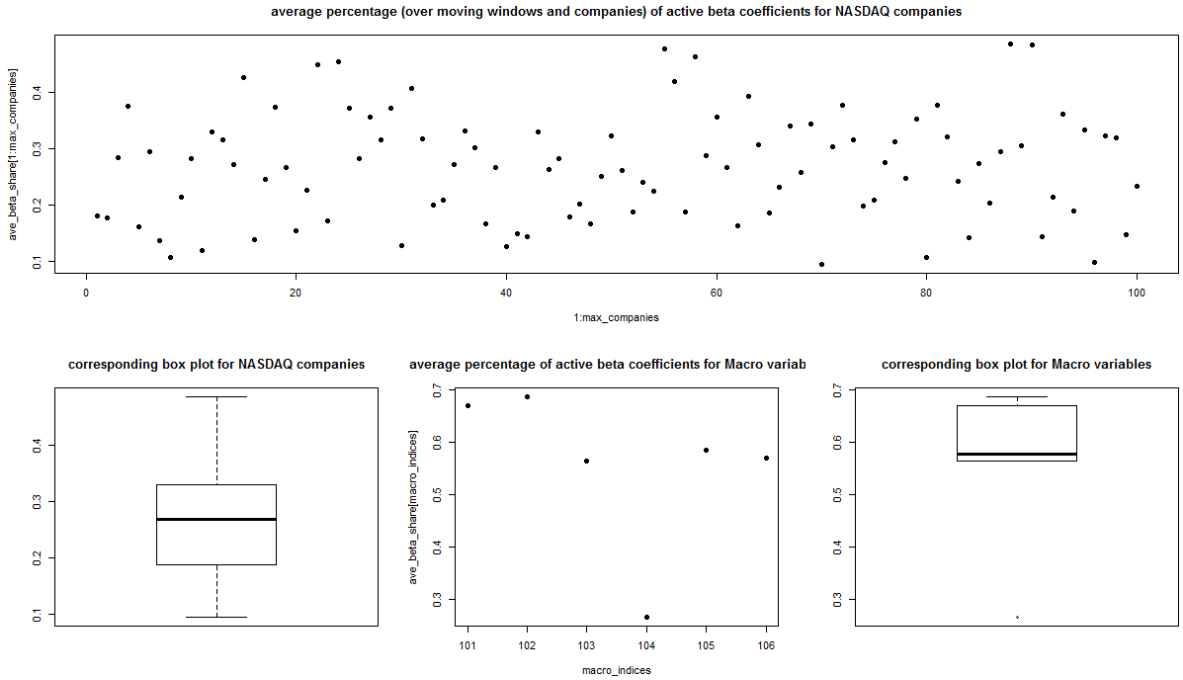

Figure 5: Average percentage (over moving windows and companies) of active beta coefficients for NASDAQ companies and macro variables and the corresponding box plots, produced via *QR.beta.stats*

The functions `QR.regressors.stats` and `QR.beta.stats` analyze the structure of the beta coefficients from the QR process. `QR.regressors.stats` provides the frequency of the covariates for a given percentage threshold `sel_threshold` and the filter value `min_regressed_comp`. For instance, for a given `sel_threshold` = 0.55 and `min_regressed_comp` = 10, we see in the first part of Listing 3 that only the following covariates: 88, 101, 102, 103, 105, 106 (88 is the number of a NASDAQ company, numbers higher than 100 are macro variables) have non-zero beta coefficients in the QR of a company. Additionally, we have the restrictions that the filtered and displayed covariates

are active regressors in at least 55% of all moving windows (n_w) for at least 10 companies. For `sel_threshold` = 0.55 and `min_regressed_comp` = 10 we can conclude that the company with the number 88 is an active regressor for some 11 companies, being present in at least 55% of all moving windows for each of those 11 companies. The macro variable with the number 101 (VIX), on the other hand, is an active regressor with non-zero beta's in all 100 NASDAQ companies, being present in at least 55% of all moving windows for each of them. The second most influential regressor is the macro variable with the number 102 (S&P500), it is an active regressor for 99 NASDAQ companies (in at least 55% of all moving windows).

```
#-----------------------------------------------
# QR.analytics: QR.regressors.stats
#-----------------------------------------------
sapply( c(0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.72), function(x) { QR.regressors.stats(
    parResult, sel_threshold = x, min_regressed_comp = 10 )$R_tab_min_regressed_comp
    })

[[1]]
 15  22  24  31  55  56  58  63  67  81  88  90 101 102 103 105 106
 31  46  51  11  74  21  55  15  10  11  79  73 100 100 100 100 100

[[2]]
 22  24  55  58  88  90 101 102 103 105 106
 13  15  23  21  43  37 100 100 93 100 98

[[3]]                        [[4]]
 88 101 102 103 105 106     101 102 103 105 106
 11 100 99  67  86  72       98  99  15  30  16

[[5]]         [[6]]        [[7]]
101 102       101 102      102
 76  87        14  38       16

#-----------------------------------------------
# QR.analytics: QR.beta.stats
#-----------------------------------------------
ave_beta_share = QR.beta.stats(parResult)
which(ave_beta_share > 0.5)
[1] 101 102 103 105 106
which(ave_beta_share > 0.666)
[1] 101 102

#-----------------------------------------------
# QR.analytics: QR.variance.vs.beta
#-----------------------------------------------
variance_vs_beta = QR.variance.vs.beta(final_data, ave_beta_share)
# truncated output for illustration
$corr_comp_vars_beta
[1] 0.5752723
$corr_macro_vars_beta
[1] 0.3024359
```

Listing 3: RiskAnalytics application example: QR.analytics part

Iterating through different `sel_threshold` values (c(0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.72)) via the `sapply` function, we can observe that with the increasing threshold only the macro

variables remain as active regressors for the NASDAQ companies. For `sel_threshold` = 0.7 only the macro variables 101 and 102 serve as regressors for 14 and 38 companies, respectively. Reaching `sel_threshold` = 0.72, the macro variable 101 (VIX) vanishes, what means that it is a regressor of maximally 9 companies, whereas the macro variable 102 (S&P500) is still an active regressor for some 16 companies.

While `QR.regressors.stats` provides the frequency of the covariates based on the active set of the beta coefficients, `QR.beta.stats` analyzes the beta coefficients themselves. Basically, `QR.beta.stats` calculates the average percentage (over all moving windows and companies) of active beta coefficients for the covariates. The average percentage of active beta coefficients with the value 0.2, for instance, would mean that the covariate, which has this percentage, acts as an active regressor in exactly 20% of all moving windows (n_w) averaged over all companies. The vector of the average percentage of active beta coefficients is stored in the variable `ave_beta_share` (each element corresponds to a covariate), see Listing 3. Besides the corresponding plots and box plots for the NASDAQ companies and macro variables, which are provided by `QR.beta.stats` based on `ave_beta_share` (see also Figure 5), `ave_beta_share` can be subjected to further statistical analysis. For example, `ave_beta_share` is minimal (= 0.095) for the company with the number 70 (Loews Corporation (L)) and is maximal (= 0.484) for the company with the number 88 (CBRE Group, Inc. (CBG)). The distribution of the `ave_beta_share` values of the macro variables is provided in Figure 5 and Table 2.

The statistical analysis provided by the functions `QR.regressors.stats` and `QR.beta.stats` reveals that the macro variables have a dominant effect on the regressed companies. Except the macro variable with the number 104 (3MTCM: the changes in the three-month Treasury bill rate) all other macro variables have an average percentage of active beta coefficients of at least 56%. The two most influential regressors are the variables 101 and 102 (VIX and S&P500). Averaged over all moving windows and regressed companies, VIX and S&P500 are present in around two thirds of the performed quantile regressions.

	^VIX	^GSPC	IYR	3MTCM	Yield	Credit
Variance	0.0187	0.0042	0.0032	0.0013	0.0447	0.0227
Beta_share	0.6695	0.6865	0.5635	0.2651	0.5840	0.5698

Table 2: Variances versus average percentage of active beta coefficients of the macro variables, produced via *QR.variance.vs.beta*

An interesting observation is the relationship between the variances of the covariates and the average percentages of active beta coefficients as calculated in `ave_beta_share`. The function `QR.variance.vs.beta` examines this issue. Among other details, this function delivers the correlations between the variances and the `ave_beta_share` values (0.575 for the companies and 0.302 for macro variables, see the last part of Listing 3), the corresponding plots and scatter plots in Figure 6, and the output for Table 2. In particular, the scatter plot in Figure 6 illustrates the positive correlation between the variances of the NASDAQ companies and the corresponding average percentages of active beta coefficients, i.e. companies with higher volatility are tendentially more often active regressors with non-zero beta coefficients.

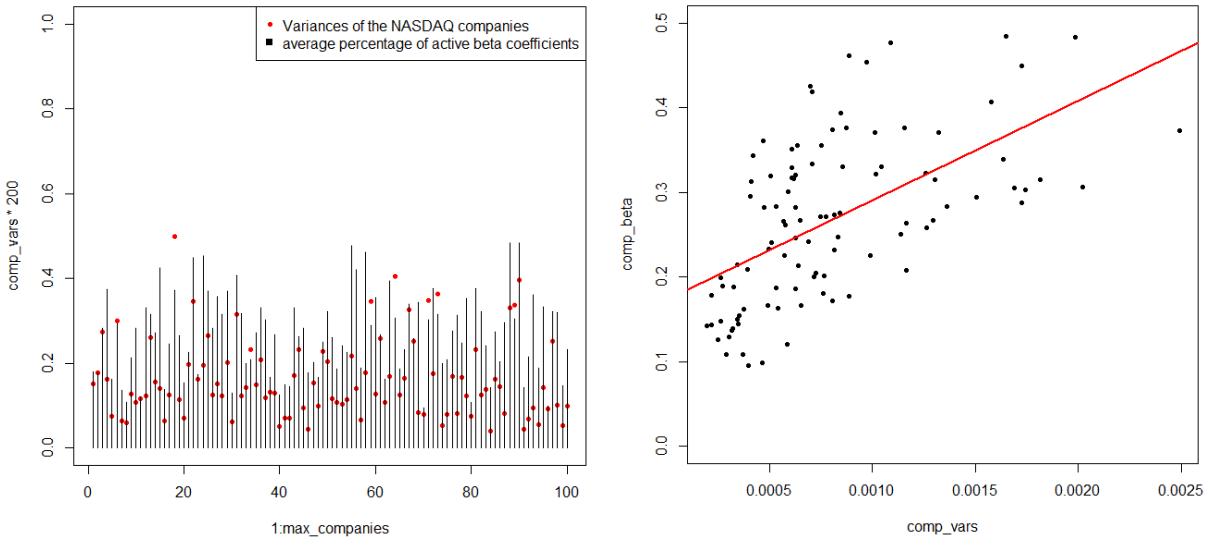

Figure 6: Variances versus average percentage of active beta coefficients of the NASDAQ companies: as a multiple plot with rescaled variances by factor of 200 on the left, and as a scatter plot with linear regression on the right, produced via *QR.variance.vs.beta*

2.4 RiskAnalytics package: "Risk Analytics" part

```
#-----------------------------------------------
# Aggregate data
#-----------------------------------------------
last_lambda = aggregate.parallel.results(final_data, max_companies, parResult,
      work_dir = work_dir, new_days = 2469, winsize = 63)

#-----------------------------------------------
# Risk Analytics
#-----------------------------------------------
lambda.analytics(last_lambda, final_data, max_companies)
# "Lambda Analytics Summary"
#     Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
# 0.004418 0.006351 0.007255 0.009206 0.009903 0.032530
# "Lambda quantiles corresponding to the given probabilities:"
#          0%         20%         40%         60%         80%        100%
# 0.004418067 0.006227005 0.006791834 0.008091999 0.010884704 0.032527493
# "last lambda value is the quantile for this probability: 0.384927066450567"
# "last lambda value: 0.00673865282581167"
# "Correlation between Lambda and macro variables"
#      ^VIX       ^GSPC         IYR      3MTCM     Yield    Credit
# 0.8196245 -0.01427044 -0.01283302 -0.0365718 0.2497622 0.6803068
```

Listing 4: RiskAnalytics application example: "Risk Analytics" part

Listing 4 shows how the QR calculation results from the parallel computing component of **RiskAnalytics**, which are saved in the `parResult` object, are aggregated and the *FRM* risk measure as proposed in the "FRM methodology and estimation" section in Yu et al. (2017) is constructed. It is recalled that the *FRM* risk measure is defined as the averaged lambda over all k NASDAQ companies:

$$FRM(t) \stackrel{def}{=} \frac{1}{k}\sum_{j=1}^{k}\lambda_j^*(t), \quad t \in \{t_0, \ldots, T\}. \tag{2}$$

The function `aggregate.parallel.results` serves the purpose of combining the λ_j^*-values from each company and applying Formula 2. Additionally, a previous lambda time series can be read in from a CSV file and concatenated with the new lambda values counting `new_days` entries. Finally, the current lambda time series is saved as a CSV file and returned as the vector `last_lambda` for further analysis.

Subsequently, the function `lambda.analytics` provides as part of the "Risk Analytics" software component descriptive statistics for the current lambda time series `last_lambda`, furthermore λ quantiles corresponding to the risk level probabilities as suggested in the "Risk levels" section in Yu et al. (2017), the last λ with its quantile probability, and the correlations between λ and the macro variables are calculated. Finally, a simple plot preview of the FRM lambda time series is generated, see Figure 7.

2.5 RiskAnalytics package: full program run

```
library(snow)
library(RiskAnalytics)

work_dir = "c:/r/frm/2017"
max_companies = 100

# Load data
companylist = get.nasdaq.companies()
system.time( yahoo_data <- get.yahoo.data(companylist, max_comp_num = max_companies,
    from_date = "2006-12-29") )
system.time( macro_data <- get.macro.data(from_date = "2006-12-28") )
final_data = combine.data(yahoo_data, macro_data)

# Calculate data
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir,
     new_days = 2469, parallel_cpu = 32, winsize = 63)
# Aggregate data
last_lambda = aggregate.parallel.results(final_data, max_companies, parResult,
      work_dir = work_dir, new_days = 2469, winsize = 63)

# Risk Analytics / QR.analytics
data.analytics(yahoo_data, macro_data)
QR.regressors.stats(parResult, sel_threshold = 0.5, min_regressed_comp = 10)
ave_beta_share = QR.beta.stats(parResult)
QR.variance.vs.beta(final_data, ave_beta_share)
lambda.analytics(last_lambda, final_data, max_companies)
```

Listing 5: RiskAnalytics application example: full program run

The full program run of the package **RiskAnalytics** is demonstrated in Listing 5. The *data processing component* extracts all needed data in real time, which are passed over to the *parallel computing component*. The latter performs the lasso penalized QR (*QR methods component*) via cluster computing (**snowfall** (Knaus, 2015)), thereby operating as an "in-memory application". That means that only the computational power of the physical CPU cores is needed and no disk Input/Output operations are required. Subsequently, the parallelization results are aggregated and the *FRM* risk measure is calculated. In conclusion, the *"Risk Analytics" component*, which comprises the tools *data.analytics*,

QR.analytics and *lambda.analytics*, provides descriptive statistics of the data collected at different stages of the **RiskAnalytics** program run, hence helping to analyze, evaluate and forecast the systemic risk for the considered markets (Nasdaq Stock Market).

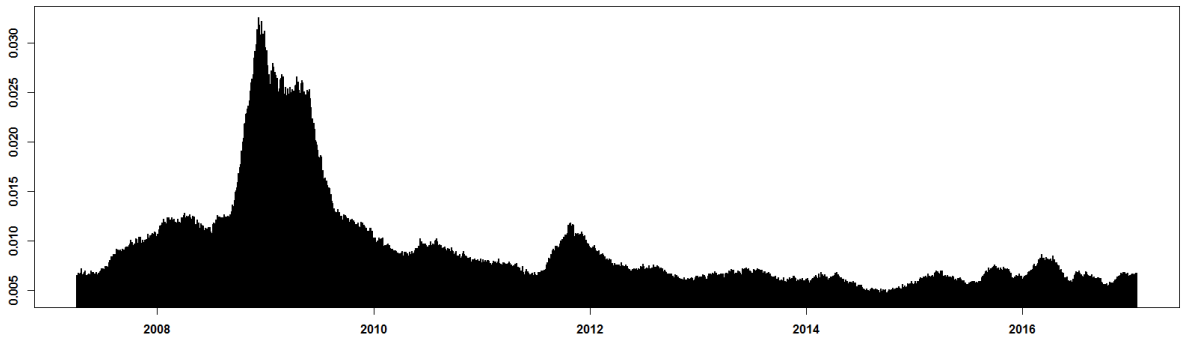

Figure 7: Simple plot preview of the FRM lambda time series, generated after the full program run of the **RiskAnalytics** package

3 RiskAnalytics (scientific IDE)

The ***RiskAnalytics*** *scientific IDE* is available under `http://borke.net/RiskAnalytics/`. IDE stands for "integrated development environment". This interactive and web based IDE has the purpose of combining and presenting the scientific, technical and visual materials, elements and sources around the topic "Risk Analytics and FRM". It provides different risk meter designs both for the risk indicators and for the time series visualizations, containing current but also previous risk measure calculations. Further, scientific references concerning the methodology but also software implementations can be found within the ***RiskAnalytics*** *scientific IDE*.

Interactive exploratory data analysis (EDA) can be conducted with the aid of the D3 (Bostock et al., 2011) based risk measure visualizations, current Google Trends statistics and real-time charts (encompassing VIX, S&P 500, Nasdaq etc.), see also Figure 8. The real-time charts are provided by TradingView, a social network for traders and investors on Stock and Futures and Forex markets (`https://www.tradingview.com/chart`).

4 Future Developments

4.1 More D3/C3 visualizations based on the beta structure

The powerful capabilities and features of the D3.js framework (Bostock et al., 2011) but also the C3.js extension, a D3-based reusable chart library (`http://c3js.org/`), can be used to implement more interactive designs and visualizations of the risk measures. For instance, the rich structure of the QR-components, lambdas and beta coefficients as time dependent vectors and matrices, can be exploited for the generation of time-variant risk dependency graphs, where the beta coefficients serve as proxies for the adjacency

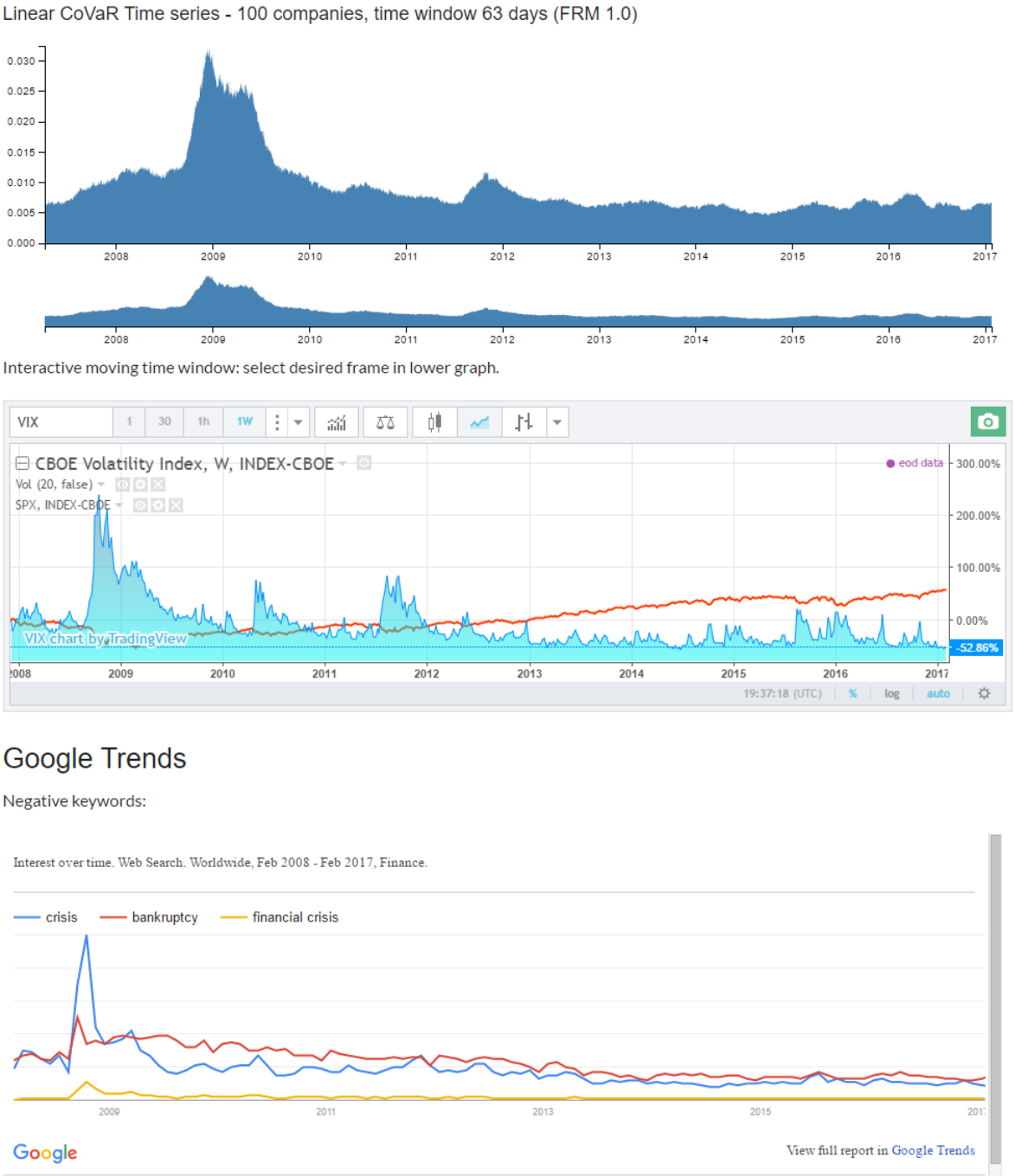

Figure 8: D3 based *FRM* risk measure visualization (created via the **RiskAnalytics** package), real-time charts (encompassing VIX and S&P 500) and current Google Trends statistics, each of them covering the same time range; available for interactive exploratory data analysis on the ***RiskAnalytics*** *scientific IDE*

matrix of the systemic risk. First steps within R can be easily done by means of the package **networkD3** (Gandrud et al., 2016), see also `https://github.com/Quantlet/forceNetwork`.

4.2 Package namespace

The current **RiskAnalytics** package could be improved by using a namespace. Namespaces make a package self-contained in two ways: the **imports** and the **exports** behavior. The **imports** defines how a function in one package finds a function in another. The **exports** helps to avoid conflicts with other packages by specifying which functions are available outside of the package (internal functions are available only within the own package and can't easily be used by another package). For more details, the following book is recommended (`http://r-pkgs.had.co.nz/namespace.html`) (Wickham, 2015). Furthermore, a package namespace could help to reduce redundant arguments, which are passed to several functions (see e.g. `parallel.lasso.computation`, `aggregate.parallel.results`), by storing the relevant variables in a namespace, from where they can be accessed from other functions without being explicitly provided as redundant arguments.

4.3 Incorporation of the *hqreg* package

The aforementioned **hqreg** (Yi, 2016) package, which provides efficient and C optimized algorithms for fitting regularization paths for lasso or elastic-net penalized regression models with Huber loss, quantile loss or squared loss, is a promising alternative for the time-intensive lasso penalized QR procedure, see Section 2.2. A further version of the **RiskAnalytics** package could provide different lasso penalized QR implementations, with **hqreg** as a possible option. But first the necessary studies and benchmarks should be carried out in order to compare the numerical consistency, reliability and time complexity with the former methods and results.

4.4 More risk measures involving the beta coefficients and the market volatility

The results from Section 2.3 indicate that there is a considerable relationship between the variances of the covariates and the average percentages of active beta coefficients, i.e. covariates with higher volatility are tendentially more often active regressors with non-zero beta coefficients. Because the L1-norm penalty in Formula (1.1) in Li and Zhu (2008) shrinks the fitted coefficients toward zero by $|\beta_1| + \ldots + |\beta_p| \leq s$, there is a duality between the λ value and the shrinkage parameter s of the β's L1-norm. Hence, the incorporation of the whole market volatility (in the given moving window or another time period) and some appropriate transformations of the β-coefficients into the new risk measure variants should be considered and examined. The ***RiskAnalytics*** *scientific IDE* is a good platform for further experiments.

5 Conclusion

The presented **RiskAnalytics** package (Borke, 2017) is a convenient tool with the purpose of integrating lasso penalized quantile regression methods with full solution paths and cluster computing support around the topic "Risk Analytics and FRM". Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and some macro variables. The derived "Risk Analytics", which comprise the methods *data.analytics*, *QR.analytics* and *lambda.analytics*, can help to forecast and evaluate the systemic risk for the corresponding markets.

Supplementary R codes are published on www.quantlet.de with the keyword FRM. The visualization and the up-to-date FRM can be found on `http://frm.wiwi.hu-berlin.de`. Additionally, the interactive and web based ***RiskAnalytics*** *scientific IDE* [1] combines the scientific, technical and visual materials, elements and sources around the research field "Risk Analytics". It is a good platform for further experiments and developments as discussed in Section 4.

References

Borke, L. (2017). *RiskAnalytics: Real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods.* R package version 0.2.0.
URL: *https://github.com/lborke/RiskAnalytics*

Bostock, M., Ogievetsky, V. and Heer, J. (2011). D3 Data-Driven Documents, *IEEE Transactions on Visualization and Computer Graphics* **17**(12): 2301–2309.
URL: *http://dx.doi.org/10.1109/TVCG.2011.185*

Gandrud, C., Allaire, J. and Russell, K. (2016). *networkD3: D3 JavaScript Network Graphs from R.* R package version 0.2.13.
URL: *https://CRAN.R-project.org/package=networkD3*

Knaus, J. (2015). *snowfall: Easier cluster computing (based on snow).* R package version 1.84-6.1.
URL: *https://CRAN.R-project.org/package=snowfall*

Koenker, R. (2016). *quantreg: Quantile Regression.* R package version 5.29.
URL: *https://CRAN.R-project.org/package=quantreg*

Koenker, R. and Mizera, I. (2014). Convex Optimization in R, *Journal of Statistical Software* **60**(1): 1–23.
URL: *https://www.jstatsoft.org/index.php/jss/article/view/v060i05*

Lang, D. T. and the CRAN team (2016). *RCurl: General Network (HTTP/FTP/...) Client Interface for R.* R package version 1.95-4.8.
URL: *https://CRAN.R-project.org/package=RCurl*

[1] `http://borke.net/RiskAnalytics/`

Li, Y. and Zhu, J. (2008). L1-Norm Quantile Regression, *Journal of Computational and Graphical Statistics* **17**(1).

Ryan, J. A. (2016). *quantmod: Quantitative Financial Modelling Framework*. R package version 0.4-7.
URL: *https://CRAN.R-project.org/package=quantmod*

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 267–288.

Tierney, L., Rossini, A. J., Li, N. and Sevcikova, H. (2016). *snow: Simple Network of Workstations*. R package version 0.4-2.
URL: *https://CRAN.R-project.org/package=snow*

Wickham, H. (2015). *R Packages*, 1st edn, O'Reilly Media, Inc.
URL: *http://r-pkgs.had.co.nz*

Yi, C. (2016). *hqreg: Regularization Paths for Lasso or Elastic-Net Penalized Huber Loss Regression and Quantile Regression*. R package version 1.3.
URL: *https://CRAN.R-project.org/package=hqreg*

Yi, C. and Huang, J. (2015). Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression, *ArXiv e-prints* . 1509.02957.

Yu, L., Härdle, W. K., Borke, L. and Benschop, T. (2017). FRM: a Financial Risk Meter based on penalizing tail events occurrence, *SFB 649 Discussion Paper* . Humboldt Universität zu Berlin.

SFB 649 Discussion Paper Series 2017

For a complete list of Discussion Papers published by the SFB 649, please visit http://sfb649.wiwi.hu-berlin.de.

001 "Fake Alpha" by Marcel Müller, Tobias Rosenberger and Marliese Uhrig-Homburg, January 2017.
002 "Estimating location values of agricultural land" by Georg Helbing, Zhiwei Shen, Martin Odening and Matthias Ritter, January 2017.
003 "FRM: a Financial Risk Meter based on penalizing tail events occurrence" by Lining Yu, Wolfgang Karl Härdle, Lukas Borke and Thijs Benschop, January 2017.
004 "Tail event driven networks of SIFIs" by Cathy Yi-Hsuan Chen, Wolfgang Karl Härdle and Yarema Okhrin, January 2017.
005 "Dynamic Valuation of Weather Derivatives under Default Risk" by Wolfgang Karl Härdle and Maria Osipenko, February 2017.
006 "RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods" by Lukas Borke, February 2017.

SFB 649, Spandauer Straße 1, D-10178 Berlin
http://sfb649.wiwi.hu-berlin.de

This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 "Economic Risk".