Introduction to the `robustGarch` package (Version 0.3.0)

Echo Liu*

August 30, 2020

Contents

*Mentors: R. Douglas Martin, Dan Hanson, and Alexios Galanos

1 Introduction

The purpose of this vignette is to show how to use the `robustGarch` package to robustly fit Garch(1,1) models, defined as $x_t = \sigma_t z_t$, where z_t is i.i.d random variables with continuous density f such that $E(z_t) = 0$, $Var(z_t) = 1$, and where the conditional variance σ_t^2 are given by

$$\sigma_t^2 = \alpha_0 + \alpha_1 x_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \ \alpha_1 + \beta_1 < 1$$

for returns time series that may have either: (1) a normal or a fat-tailed non-normal Garch model innovations distribution, or (2) additive outliers. The most common way of fitting a Garch model is to compute a maximum-likelihood estimate (MLE) of the parameters based on the assumption that the innovations have a normal distribution, and when the method used for possibly non-normal innovations distributions, the method is called Quasi-Maximum Likelihood (QML) method.

Bias caused by outliers has been studied in some literature. Huber (1981) established two properties for a good robust model fitting method. The first is that a robust estimate needs to be efficient and should be efficient in the sense of having a variance that is not much larger than that of a maximum likelihood estimate typically based on a normal distribution. The second property, introduced by later researchers is that replacing a small fraction of the data by outliers should not cause big difference as measured by estimator bias. Mendes (2000) showed how outliers create asymptotic bias in Quasi Maximum Likelihood Estimate of the Garch parameters. Muler and Yohai (2008)[1] showed that QML parameter estimation based on normal likelihood is very sensitive to outliers in financial returns, and even a single outlier can have a huge influence on the QML parameter estimates. Thus, it is very crucial to build accurate robust parameter estimation methods.

The `robustGarch` package provides two main methods for fitting robust Garch processes, M-Estimates based on a bounded loss function, and so-called BM estimates that are M-estimates with a bounded rho function combined with a robust variance filtering algorithm that prevents the propagation of additive outliers. The package also contains QML method as an option for the user. It currently contains fitting and plotting methods for fitted models.

2 Using the `robustGarch` Package

This document discusses the details of the above two robust Garch(1, 1) estimator methods, and how they are implemented in the package with some examples[2].

[1] We reference the paper as MY2008 in the later content

[2] the development version on github (https://github.com/EchoRLiu/robustGarch) with examples and and demos

The `robustGarch` package can be installed as following:

```
devtools::install_github("EchoRLiu/robustGarch")

##
##
   checking for file '/private/var/folders/zw/q0qwt2310yz3p73ggyx5cwr00000gp/T/Rtmpl

v  checking for file '/private/var/folders/zw/q0qwt2310yz3p73ggyx5cwr00000gp/T/Rtmpl
##

-  preparing 'robustGarch':
##

   checking DESCRIPTION meta-information ...

v  checking DESCRIPTION meta-information
##

-  checking for LF line-endings in source and make files and shell scripts
##

-  checking for empty or unneeded directories
##

-  building 'robustGarch_0.1.0.tar.gz'
##



##

library(robustGarch)
```

To use `robGarch` function, the user needs to specify two important parameters, `methods` and `fixed_pars`, for the main fitting function `robGarch`[3].

- For BM estimators, the user needs to choose "`bounded MEst`" as `methods`, and to control the robustness, `fixed_pars = c(div, k)` (default c(0.8, 3.0)) need to be specified.

[3]`To see the complete arguments of robGarch, use args(robGarch)`

- When the `methods` parameter of `robGarch` is specified as "`modified MEst`", only `div` (default = 0.8) controls the robustness of M estimators. The user should specify the `methods` as "`modified MEst`" and `fixed_pars = div`.

- QML is obtained without any restriction or control over the outlier effect. The user only needs to choose the "`QML`" as `methods` to specify the estimator.

With the `methods` and `fixed_pars` parameters specified, the `robGarch` function can take in the `data`, and output the parameter estimations. The following is an example using the internal data in `robustGarch` package.

```
data(rtn)
fit <- robGarch(rtn[1:604], methods="bounded MEst", fixed_pars=c(0.8, 3.0))

##
## Iter: 1 fn: 2.7158  Pars:  0.04251 0.17465 0.82392 0.10650
## Iter: 2 fn: 2.7158  Pars:  0.04250 0.17463 0.82394 0.10646
## solnp--> Completed in 2 iterations
##      alpha_0      alpha_1       beta_1
## 8.647505e-06 1.746338e-01 8.239391e-01

summary(fit)

## Model:  bounded MEst
## with div =  0.8 , k =  3
## Observations:  604
##
## Result:
##                     alpha_0   alpha_1       beta_1
## fitted_pars    8.647505e-06 0.1746338 8.239391e-01
## standard_error 2.002408e-01 0.3144657 1.335392e-01
## t_value        2.122592e-01 0.5553348 6.170017e+00
## p_value        8.319048e-01 0.5786657 6.828247e-10
##
## Log-likelihood:  2.715769
##
## Optimizer:  Rsolnp
##
##
## Time elapsed:  1.491095
## Convergence Message:  0
```

2.1 Some Examples and Data Testing Results

The following sub-sections provide the results of testing `robustGarch` package function `robGarch` and comparing the results with those obtained using the `rugarch` package function `ugarchfit` for the following six data sets:

- SP500 daily returns from February 1, 2000 through June 30, 2002, a total of n = 604 observations
- 5 artificially generated Garch(1, 1) returns, a total of n = 500 observations, using the five estimators in Table 1 that were provided by Doug Martin[4]

	μ_e	σ	α_0	α_1	β_1	$\alpha_1 + \beta_1$
1	0.01	0.07	8.76e-04	0.135	0.686	0.821
2	0.01	0.07	3.90e-04	0.144	0.776	0.920
3	0.01	0.07	2.43e-04	0.117	0.833	0.950
4	0.01	0.07	2.45e-04	0.103	0.867	0.971
5	0.01	0.07	1.01e-4	0.083	0.909	0.992

Table 1: Garch(1,1) Parameter Values Used for robustGarch Test

We are testing the following three Garch(1, 1) fitting methods from the `robustGarch` package:

1. QML method: quasi-maximum likelihood method (MLE for normal distribution Garch(1, 1) innovations)
2. M estimator: a bounded loss function M-estimator, as described in MY2008, with tuning parameter $div = 1.0$
3. BM estimator: M1 with robust filter added, with $div = 1.0$ and filter tuning parameter $k = 5.02$

In further testing we will also include M2 and BM2, which are more robust versions of M and BM, obtained by using smaller values of div and k.

2.1.1 Test Result with SP500 Returns

For the SP500 returns we also compare our `robustGarch` results with those in MY2008, and refer to their results as the MY results. Table 2 below contains the results of testing our QML, M1, BM1 methods, along with the MY results labeled QML-MY, BM1-MY (MY results did

[4]For more details, please refer to Chen and Martin (2017)

not provide M1 result), in comparison with the `rugarch` results (in the first row) using the SP500 data.

The main conclusion of Table 2 for our estimators QML, M1 and BM1 is that the point estimates seem reasonable. We are currently working on the standard error, t stats and p stats, which would be available for the users in the future.

Table 2: Analysis for Garch(1,1) models with SP500 data (02/01/2000-06/30/2002)

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
rugarch	1.1e-05	0.110	0.84	0.95
QML	9.0e-06	0.095	0.87	0.96
QML-MY	3.9e-06	0.110	0.86	0.97
M1	7.2e-06	0.087	0.88	0.96
BM1	7.2e-06	0.087	0.88	0.96
BM1-MY	4.5e-06	0.100	0.84	0.94

Comparing Figure 2 with Figure 1, we can notice that the BM1 captains the main dynamics of the SP500 series, but has smaller lagging, which shows BM estimator is more robust over outliers.

```
library(rugarch)
spec <- ugarchspec(mean.model = list(armaOrder = c(0,0), include.mean = FALSE))
mod_rugarch <- ugarchfit(spec, rtn[1:604])
plot(mod_rugarch, which=3)
```

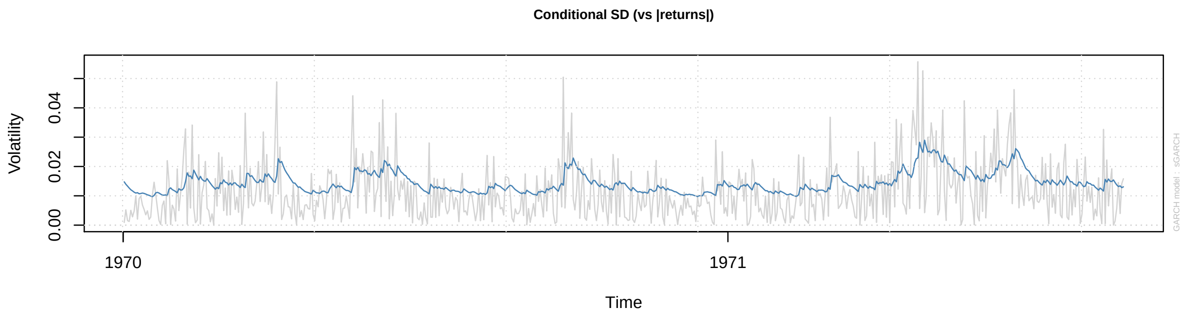

Figure 1: Conditional SD vs |returns| of SP500 with rugarch

```
fit_BM1 <- robGarch(rtn[1:604], methods = "bounded MEst", fixed_pars = c(1.0, 5.02)

##
## Iter: 1 fn: 2.3083  Pars:  0.03558 0.08653 0.87511 0.08805
## Iter: 2 fn: 2.3083  Pars:  0.03558 0.08654 0.87511 0.08806
## solnp--> Completed in 2 iterations
##       alpha_0      alpha_1       beta_1
## 7.239803e-06 8.653593e-02 8.751077e-01

plot(fit_BM1, main_name = "Conditional SD (vs |returns|) of SP500 with BM1 - robust
```

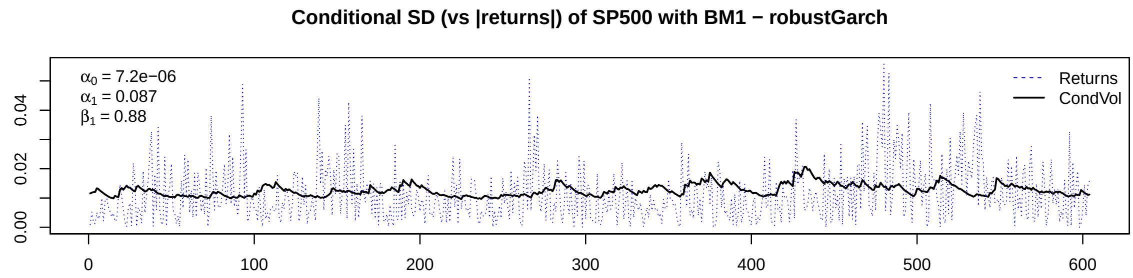

Figure 2: Conditional SD vs |returns| of SP500 with robustGarch BM1 (div = 1.0, k = 5.02)

2.1.2 Test Result with Simulated Garch Series

In Table 3 through Table 7, for returns generated by the five Garch(1, 1) models in Table 1, we present the `rugarch` results along with the results of our QML, M1 and BM1 methods. We note that the parameter estimates of QML are reasonably close to those of `rugarch` in Table 3-7, which shows the parameter estimation of `robustGarch` package is correct.

Table 3: Garch(1,1) model fits for returns simulated with first model, $\alpha_1 + \beta_1 = 0.821$)

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
true parameters	0.00088	0.14	0.69	0.82
rugarch	0.00072	0.19	0.65	0.84
QML	0.00072	0.19	0.65	0.84
M1	0.00075	0.11	0.69	0.81
BM1	0.00077	0.14	0.67	0.81

Similar to SP500 test results, from Figure 3 to Figure 12, the conditional standard deviation is more robust in BM1 compared to rugarch.

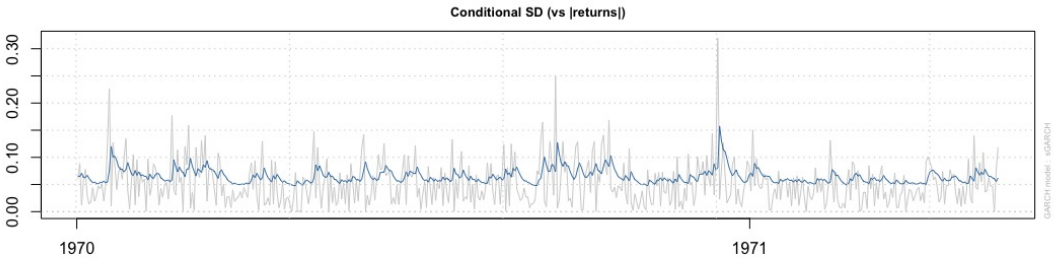

Figure 3: Conditional SD vs |returns| of first model with rugarch

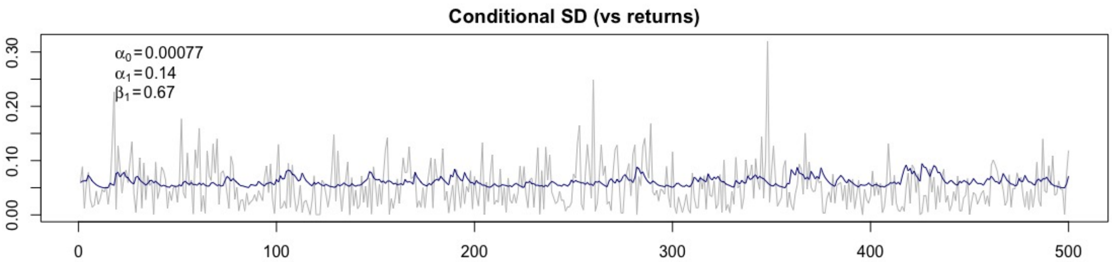

Figure 4: Conditional SD vs |returns| of first model with robustGarch BM1

Table 4: Garch(1,1) model fits for returns simulated with second model, $\alpha_1 + \beta_1 = 0.920$

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
true parameters	0.00039	0.14	0.78	0.92
rugarch	0.00082	0.12	0.66	0.78
QML	0.00068	0.11	0.72	0.82
M1	0.00075	0.12	0.70	0.82
BM1	0.00075	0.12	0.69	0.82

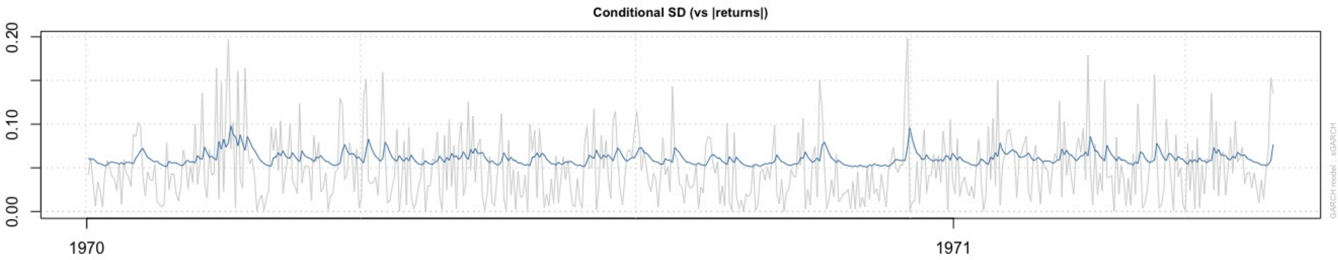

Figure 5: Conditional SD vs |returns| of second model with rugarch

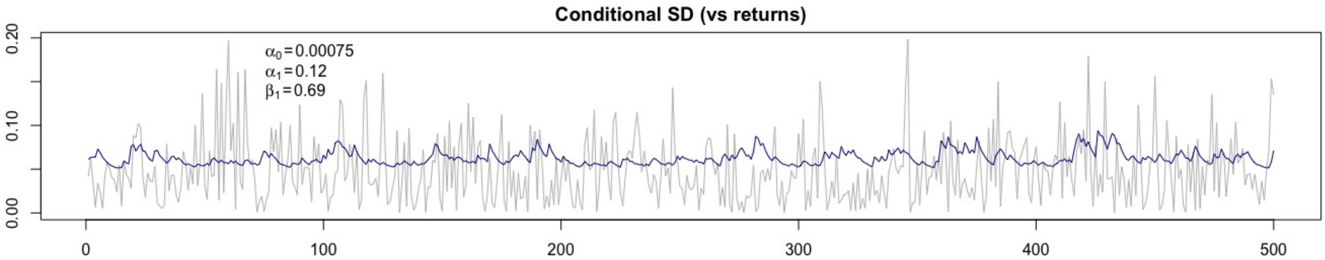

Figure 6: Conditional SD vs |returns| of second model with robustGarch BM1

Table 5: Garch(1,1) model fits for returns simulated with third model, $\alpha_1 + \beta_1 = 0.950$

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
true parameters	0.00024	0.120	0.83	0.95
rugarch	0.00058	0.082	0.76	0.84
QML	0.00037	0.061	0.84	0.90
M1	0.00038	0.067	0.83	0.90
BM1	0.00037	0.071	0.84	0.91

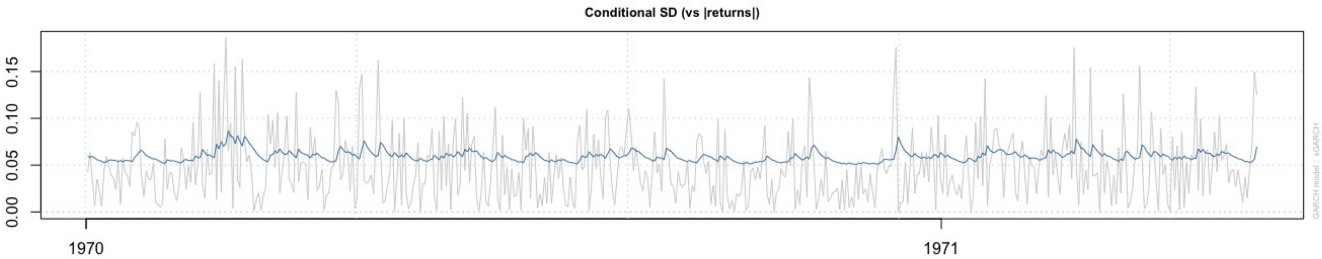

Figure 7: Conditional SD vs |returns| of third model with rugarch

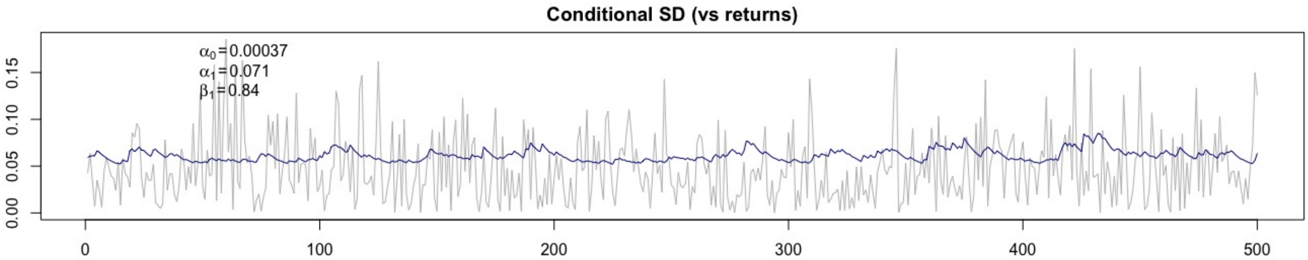

Figure 8: Conditional SD vs |returns| of third model with robustGarch BM1

Table 6: Garch(1,1) model fits for returns simulated with fourth model, $\alpha_1 + \beta_1 = 0.971$

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
true parameters	0.00024	0.100	0.87	0.97
rugarch	0.00051	0.073	0.85	0.93
QML	0.00057	0.078	0.84	0.92
M1	0.00058	0.087	0.83	0.92
BM1	0.00058	0.087	0.83	0.92

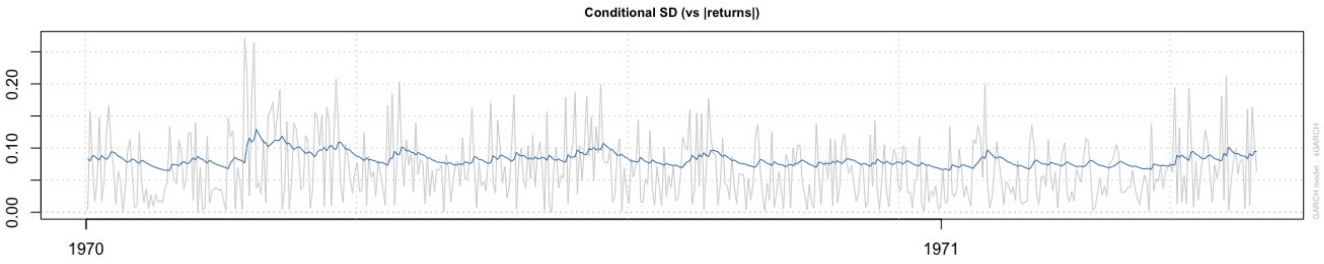

Figure 9: Conditional SD vs |returns| of fourth model with rugarch

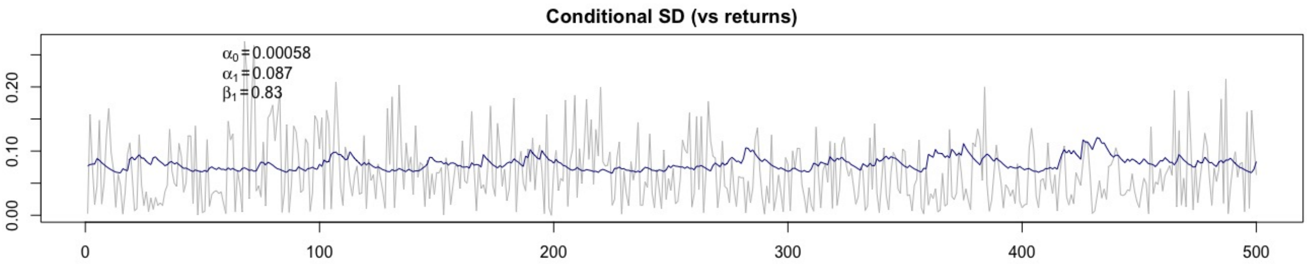

Figure 10: Conditional SD vs |returns| of fourth model with robustGarch BM1

Table 7: Garch(1,1) model fits for returns simulated with fifth model, $\alpha_1 + \beta_1 = 0.992$

	α_0	α_1	β_1	$\alpha_1 + \beta_1$
true parameters	1.0e-04	0.083	0.91	0.99
rugarch	9.3e-05	0.041	0.95	0.99
QML	1.0e-04	0.043	0.95	0.99
M1	1.1e-04	0.048	0.94	0.99
BM1	1.1e-04	0.048	0.94	0.99

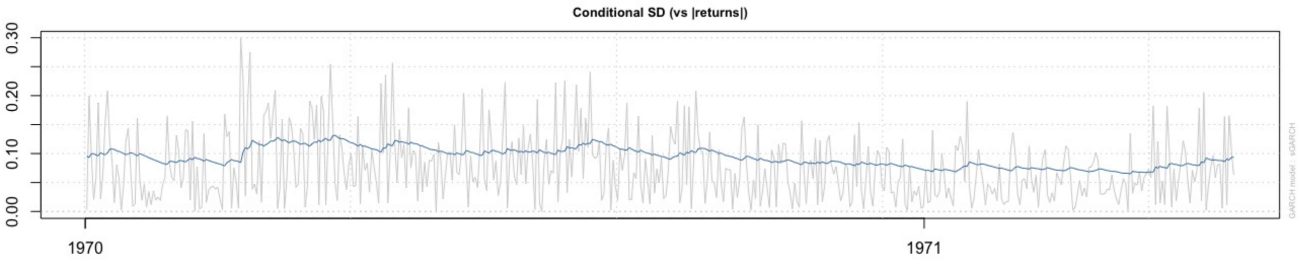

Figure 11: Conditional SD vs |returns| of fifth model with rugarch

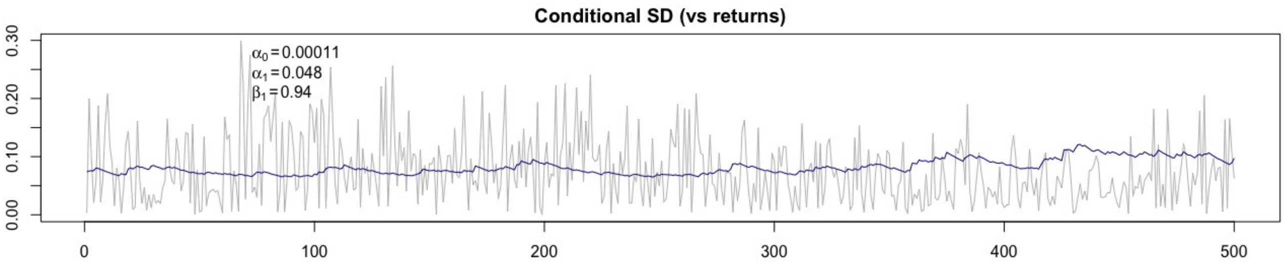

Figure 12: Conditional SD vs |returns| of fifth model with robustGarch BM1

3 Future Development

Any future development will be released in the github page. A few key features will be added to the package in September 2020:

- Fix the issue with singularity error with Hessian matrix
- Statistics tests such as std_error, t_value, p_value for Garch parameters
- Code debug on model filter for M model and QML
- More optimization choices

- Extension to robust Garch(p, q)
- Extension to Windows system

References

Chen, X. and Martin, R. (Jan. 2017). “Standard Errors of Risk and Performance Estimators with Serially Correlated Returns”. In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3085672.

Huber, P. J. (1981). *Robust statistics*. Vol. 523. John Wiley Sons.

Mendes, B. M. (2000). “Assessing the bias of maximum likelihood estimates of contaminated GARCH Models”. In: *Journal of Statistical Computation and Simulation* 67, pp. 359–376.

Muler, N. and Yohai, V. J. (2008). “Robust estimates for GARCH models”. In:

A BM Estimators

Robust Garch is first introduced by Muler and Yohai (2008) with the foundation work of Huber (1981). Defined as $x_t = \sigma_t z_t$, where z_t is i.i.d random variables with continuous density f such that $E(z_t) = 0$, $Var(z_t) = 1$, and where the conditional variance σ_t^2 are given by

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i x_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{1}$$

the robust parameters estimation is based on M-estimate with more flavors.

The main structure of parameter estimation is

$$\hat{\gamma_T}^B = \begin{cases} \hat{\gamma_{1,T}}, & M_T(\hat{\gamma_{1,T}}) \le M_{Tk}^*(\hat{\gamma_{2,T}}) \\ \hat{\gamma_{2,T}}, & M_T(\hat{\gamma_{1,T}}) > M_{Tk}^*(\hat{\gamma_{2,T}}) \end{cases} \tag{2}$$

where

$$\begin{aligned} \hat{\gamma_{1,T}} &= arg \min_{\mathbf{c}} M_T(\mathbf{c}) \\ &= arg \min_{\alpha_i, \beta_j} \frac{1}{T-p} \sum_{t=p+1}^{T} \rho(log(x_t^2) - log(\alpha_0 + \sum_{i=1}^{p} \alpha_i x_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2)) \end{aligned} \tag{3}$$

and

$$
\begin{aligned}
\hat{\gamma_{2,T}} &= arg \min_{\mathbf{c}} M^*_{Tk}(\mathbf{c}) \\
&= arg \min_{\alpha_i,\beta_j} \frac{1}{T-p} \sum_{t=p+1}^{T} \rho(log(x_t^2) \\
&- log(\alpha_0 + \sum_{i=1}^{p} \alpha_i \sigma^2_{t-i,k}(\alpha_i,\beta_j) r_k(\frac{x^2_{t-i}}{\sigma^2_{t-i,k}}) + \sum_{j=1}^{q} \beta_j \sigma^2_{t-j}))
\end{aligned} \tag{4}
$$

where

$$
r_k(u) = \begin{cases} u, & u \leq k \\ k, & u > k \end{cases} \tag{5}
$$

The use of Equation 5 is to restrict the propagation of the outlier effect, and thus small **k** can provide a robust model. The different optimization under different cases makes sure that the model has robustness with outliers and maintains consistency when the series follows a Garch without outliers.

BM estimators are also controlled by $\rho = m_1(\rho_0)$, where $\rho_0 = -log(\frac{1}{\sqrt{2\pi}} e^{-(e^w - w)/2})$, $w = log(z_t^2)$, and m_1 is a non-decreasing, bounded function, defined as

$$
m_1(x) = \begin{cases} x, & x \leq 4 \\ P(x), & 4 < x \leq 4.34.15, x > 4.3 \end{cases} \tag{6}
$$

and

$$
\begin{aligned}
P(x) = \frac{2}{(b-a)^3}(\frac{1}{4}(x^4 - a^4) - \frac{1}{3}(2a+b)(x^3 - a^3) + \frac{1}{2}(a^2 + 2ab)(x^2 - a^2)) \\
- \frac{2a^2 b}{(b-a)^3}(x-a) - \frac{1}{3(b-a)^2}(x-a)^3 + x, a = 4, b = 4.3
\end{aligned} \tag{7}
$$

satisfying constraints $P(a) = a, P'(a) = 1, P'(b) = P''(a) = P''(b) = 0$. To have different level of robustness, BM estimators can use $\rho = div * m_1(\rho_0/div)$, where $0 < div <= 1$. The smaller div is, the larger ρ_0/div would be, thus obtaining more control and gaining more robustness.

B M Estimators

M estimators are similar to BM estimators without the restriction on the propagation of the outlier effect. If k in Equation 5 is large, then $\sum_{i=1}^{p} \alpha_i \sigma^2_{t-i,k}(\alpha_i,\beta_j) r_k(\frac{x^2_{t-i}}{\sigma^2_{t-i,k}})$ becomes the normal form $\sum_{i=1}^{p} \alpha_i x^2_{t-i}$, and thus lose the robustness of BM estimators.