Modeling and Forecasting the Volatility of US Bond Yields Using GARCH Model

Lily (Lizheng) Zhou

December 19, 2019

Abstract

In finance, people are interested in the volatility, or variability, of time series. This is because volatility is an obvious choice to measure risk. This project did a series of standard modeling and forecasting processes on the volatility of US bond yields using GARCH model. From those processes, we fitted the model, chose the best model, also did forecasting. The best model chosen performances quite well. From the project, we also found some properties of volatility structure and further research questions and steps can do.

1 Introduction

In finance, the yield curve is a curve showing several yields or interest rates across different contract lengths (1 month, 1 year, 10 year, 20 year, etc.) for a similar debt contract. The curve shows the relation between the (level of the) interest rate (or cost of borrowing) and the time to maturity, known as the "term", of the debt for a given borrower in a given currency.

The U.S. Treasury Department issues bonds with maturities ranging from one month to 30 years. As bonds with longer maturities usually carry higher risk, such bonds have higher yields than do bonds with shorter maturities. Due to this, a normal yield curve reflects increasing bond yields as maturity increases.

In financial markets, where we measure prices frequently, volatility (which is analogous to standard deviation) is an obvious choice to measure risk. But in real markets, volatility changes with the market itself. When modeling financial time series, we often consider log return of time series x_t, which we denote it as y_t:

$$y_t = \bigtriangledown log(x_t) = log(x_t) - log(x_{t-1})$$

I will both plot differenced and log returns of series later in this project.

The historical yield data are published by the US Federal Reserve Data Releases and imported from Quandl (https://www.quandl.com/data/FED/SVENY), I used the US Bonds from 6/14/1961 to 12/13/2019 with 1-30 Year Maturities.

These yield curves are an off-the-run Treasury yield curve based on a large set of outstanding Treasury notes and bonds, and are based on a continuous compounding convention. Values are daily estimates of the yield curve from 1961 for the entire maturity range spanned by outstanding Treasury securities.

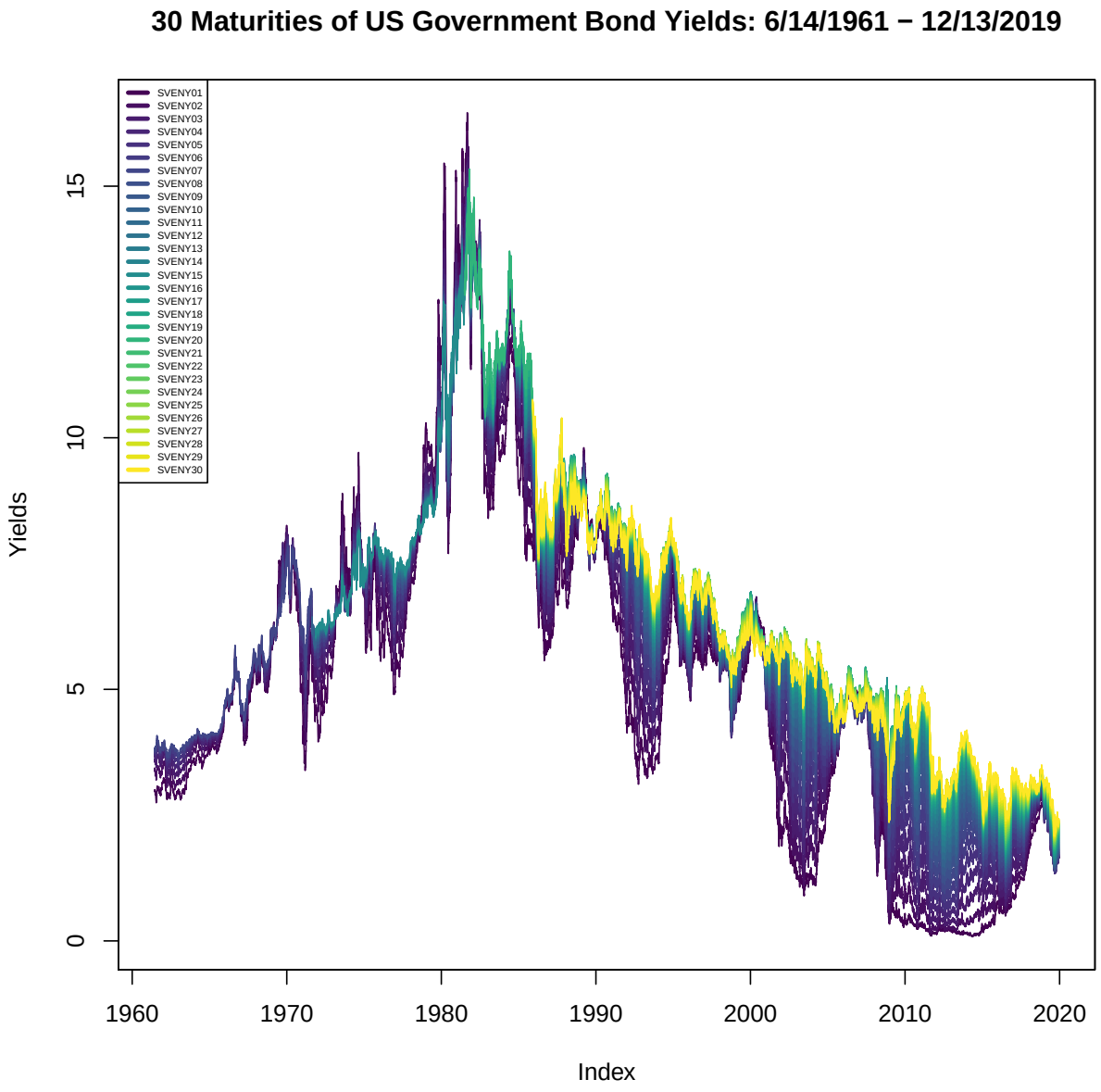
2 Exploratory Data Analysis

2.1 Data Analysis of the Full Data

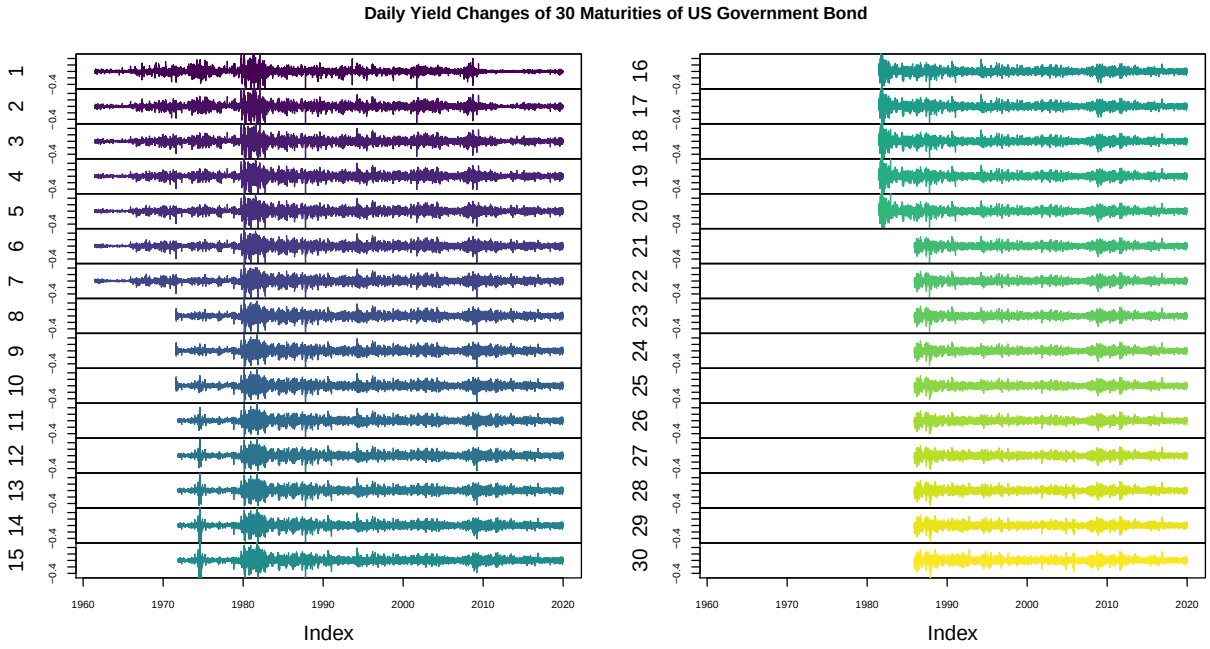
We first look at the whole evolution of bond yields from 6/14/1961 to 12/13/2019 for whole 30 maturities.

These data include the whole yield curve. The yield of a bond is the price of the money lent. The higher the yield, the more money you receive on your investment. The yield curve has many maturities; in this dataset, it ranges from 1 year to 30 years. Different maturities have different yields, but yields of neighboring maturities are relatively close to each other and also move together.

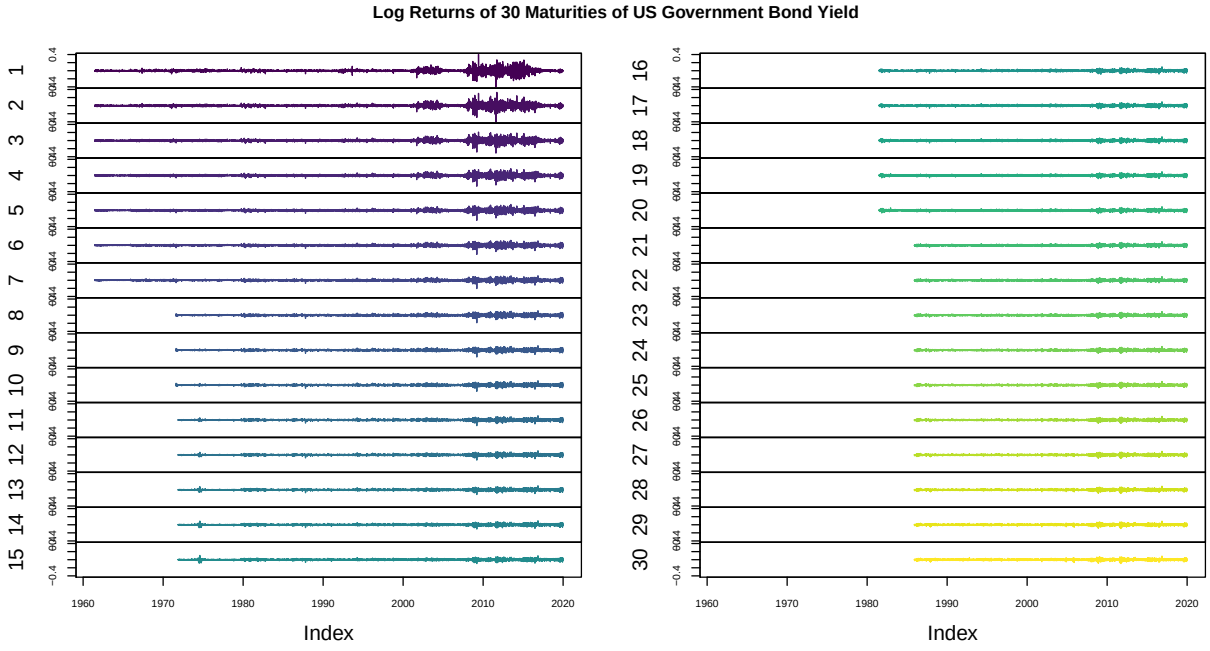
Let's visualize the yields over time. We will see that the long yields (e.g. SVENY30) tend to be more stable in the long term, while the short yields (e.g. SVENY01) vary a lot. These movements are related to the monetary policy of the FED and economic cycles.



Currently, we are at yield levels, it is obvious that the raw data of bond yields for 30 matuurities are non-stationary due to the upward trend and non-constant variance. We need to calculate the daily changes in the yield levels. This is called "differentiation" in time series analysis. Differentiation has the added benefit of making a time series independent of time.



We can also plot the daily log returns:



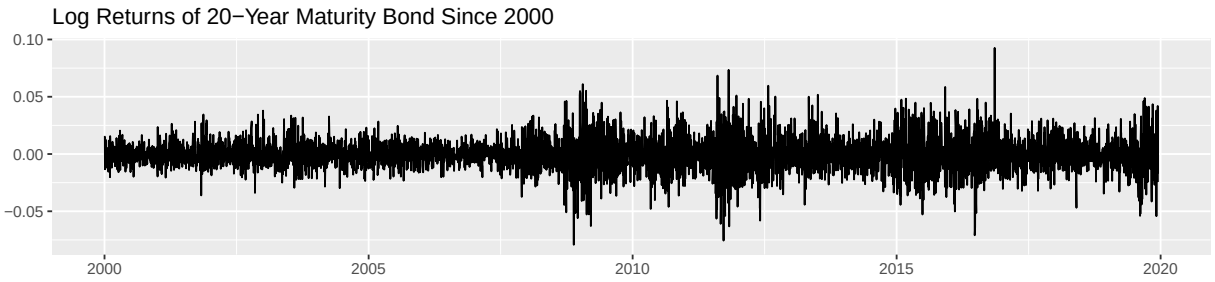
By taking a look at the time series from the previous plots, we see hints that the returns following each other have some unique properties:

The **direction** (positive or negative) of a return is mostly independent of the previous day's return. In other words, you don't know if the next day's return will be positive or negative just by looking at the time series. The **magnitude** of the return is similar to the previous day's return. That means, if markets are calm today, we expect the same tomorrow. However, in a volatile market (crisis), you should expect a similarly turbulent tomorrow.

2.2 Data Analysis of the Log Return of Bond Yield for Only 20-Year Maturity Since 2000

There are 30 maturities of bond yields and from a long time ago. I only want to focus on one single maturity bond, in this project I will only analyse 20-year maturity bond; because we are interested in the recent evolution of bond yields, we will filter the time series for data from 2000 onward.

Since we analyse the volatility based on the log returns of yield bond for only 20-year maturity since 2000, we plot the sub-dataset we focus on:



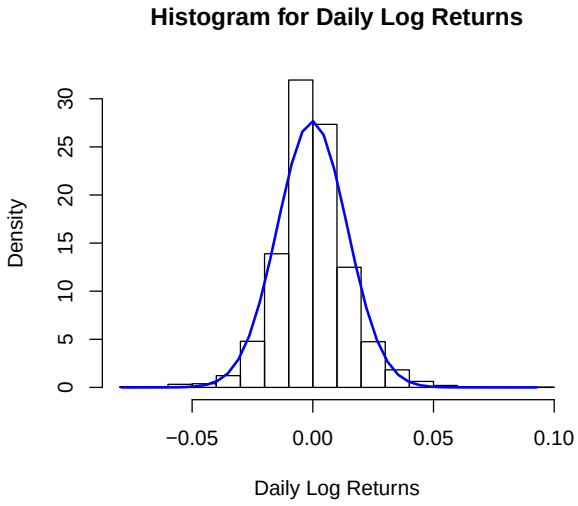
To test if the log returns of yield bond for only 20-year maturity since 2000 data is stationary, use `adf.test()` function.

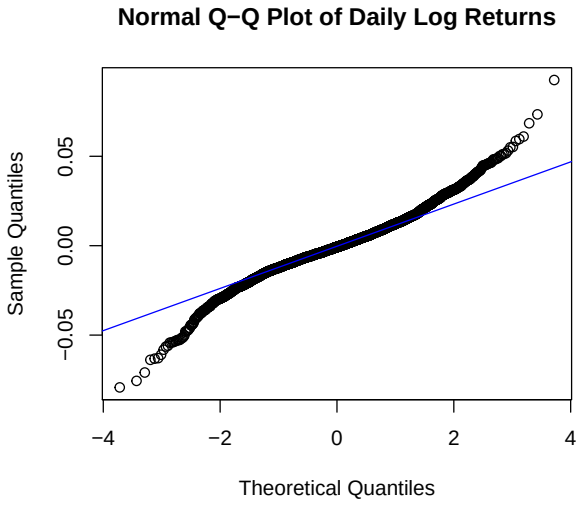
```
##
##  Augmented Dickey-Fuller Test
##
## data:  x_ld_20
## Dickey-Fuller = -15.726, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

From the result, we see that Augmented Dickey-Fuller test confirms the above finding by allowing us to reject the null hypothesis of a unit root and accept the alternative hypothesis that **the series is stationary** with p-value is 0.01, which is less than 0.05.

Then, we compute the mean of the log return data, got μ=$-2.2521358 \times 10^{-4}$, which is very small; and did a two-sided t-test to confirm that this series mean is **not significantly different from 0**.

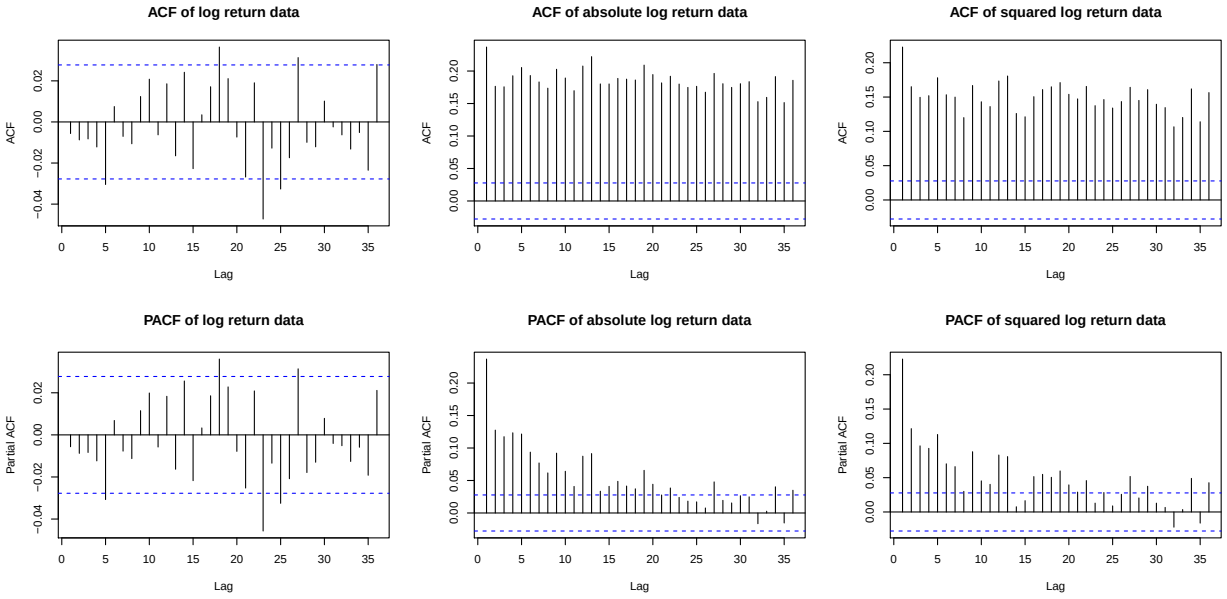
Also look at the distribution of log return data:



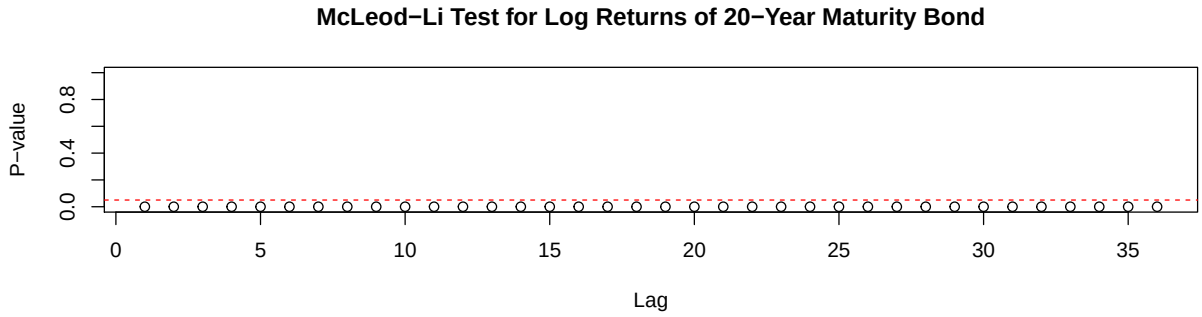


As seen from the histogram and the QQ-plot, the series has a somewhat normal distribution with fat tails at both ends.

Now, we look at the ACF and PACF of log returns, absolute log and squared log returns plots.



When look at plots of the ACF and PACF of the log return data, they don't show significant except limited lags, thus we see that the data are uncorrelated over time. However, both the squared and the absolute log return values have high correlation. Thus, we may conclude that the log returns process has a strong non-linear dependence.



The McLeod-Li test suggests significant ARCH effects in the log return series.

3 Model Fitting and Selection

As was shown ealier, the plots of the ACF and PACF of the return data show show no clear significant, so the we cannot conclude any ARMA model we can use. Also, when use `auto.arima` function to check the log return series:

We see that we still got ARIMA(0,0,0) with zero mean suggestion. Thus, I decided to use ARIMA(0,0,0) model, then only need to compare different GARCH models.

Fit several models shown below, compare the AIC and BIC of them:

Model	AIC	BIC
GARCH(1,0)	-5.7126	-5.71
GARCH(1,1)	-5.8941	-5.8902
GARCH(1,2)	-5.8945	-5.8893
GARCH(2,1)	-5.8938	-5.8885
GARCH(2,2)	-5.8941	-5.8876

we notice that AIC and BIC both are very close, respectively. Because ARCH(1) model has relative large AIC and BIC, while other models' AIC, BIC are extremely close, I prefer simplest model when excluded the ARCH(1), which means we choose GARCH(1,1). Also, if simply choose the smallest BIC here, the best model is still GARCH(1,1) of log return data.

When choose conditional distribution, we can assume the errors are normally distributed, we could also assume that they have a t-distribution also for fatter tail; also adding skewed or not-skewed into consideration, I fit the GARCH(1,1) with 4 conditional distributions: normal distibution ("norm"), skew-normal distribution ("snorm"), student-t distribution ("std") and skew student-t distribution ("sstd").

Model	AIC	BIC
GARCH(1,1) with "norm"	-5.8941	-5.8902
GARCH(1,1) with "snorm"	-5.8953	-5.8901
GARCH(1,1) with "std"	-5.907	-5.9018
GARCH(1,1) with "sstd"	-5.9097	-5.9031

We see that GARCH(1,1) with skew student-t distribution has smallest AIC and BIC, which is also more realistic approach in real world. **I will choose skew student-t distribution** here according to smallest AIC and BIC.

Actually, when doing the adjusted pearson goodness-of-fit test for 4 distribution methods with `rugarch` package:

Table 3: Goodness-of-Fit Test with "norm"

group	statistic	p-value(g-1)
20	46.12127	0.0004764
30	57.83704	0.0011387
40	71.53317	0.0011420
50	75.01082	0.0098156

Table 4: Goodness-of-Fit Test with "snorm"

group	statistic	p-value(g-1)
20	32.47525	0.0276114
30	43.41732	0.0416407
40	55.83464	0.0393350
50	68.15574	0.0364279

Table 5: Goodness-of-Fit Test with "std"

group	statistic	p-value(g-1)
20	40.73341	0.0026186
30	44.81239	0.0306823
40	54.90459	0.0469744
50	71.84386	0.0183908

Table 6: Goodness-of-Fit Test with "sstd"

group	statistic	p-value(g-1)
20	19.95971	0.3970062
30	36.77871	0.1519694
40	34.04269	0.6951910
50	39.21207	0.8401061

The adjusted pearson goodness-of-fit test compares the empirical distribution of the standardized residuals with the selected theoretical distribution. The null hypothesis is that the empirical and theoretical distribution is identical.

We see that only "sstd" has all p-values for group 20, 30, 40, 50 are all larger than 0.05, hence we fail to reject null hypothesis, and we also will choose skew student-t distribution ("sstd").

4 Model Estimation

As shown earlier, the mean of series is constant and is not significantly different from 0, we assume it as 0 and didn't include mean when fitting the model. Using `rugarch` package, we fit GARCH(1,1) with skew student-t distribution ("sstd") and get the following estimation for the parameters:

I chose to look at robust errors here.

Table 7: Estimation Results (Robust Standard Error)

	Estimate	Std. Error	t value	Pr(>|t|)
omega	0.0000005	0.0000031	0.1711115	0.8641361
alpha1	0.0438807	0.0127763	3.4345473	0.0005935
beta1	0.9547189	0.0122986	77.6279480	0.0000000
skew	1.0814622	0.0227805	47.4731918	0.0000000
shape	11.4288748	4.7379389	2.4122039	0.0158564

The GARCH(1,1) models log returns as:

$$y_t = \sigma_t \epsilon_t,$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

All estimated parameters are statistically significant except α_0, while α_0=5.274691×10^{-7} is extremely small, won't affect the model too much, so we think it is fine.

Thus, we get the GARCH(1,1) with skew student-t distribution ("sstd") is: $y_t = \sigma_t \epsilon_t$, with $\sigma_t^2 = 5.274691 \times 10^{-7} + 0.0438807\ y_{t-1}^2 + 0.9547189\ \sigma_{t-1}^2$, and $\epsilon_t \sim$ *iid N(0,1).*

5 Diagnostic Checking

$\alpha_1 + \beta_1 = 0.9985996 < 1$, thus the stationarity condition hold.

Then, look at the test of indenpendence:

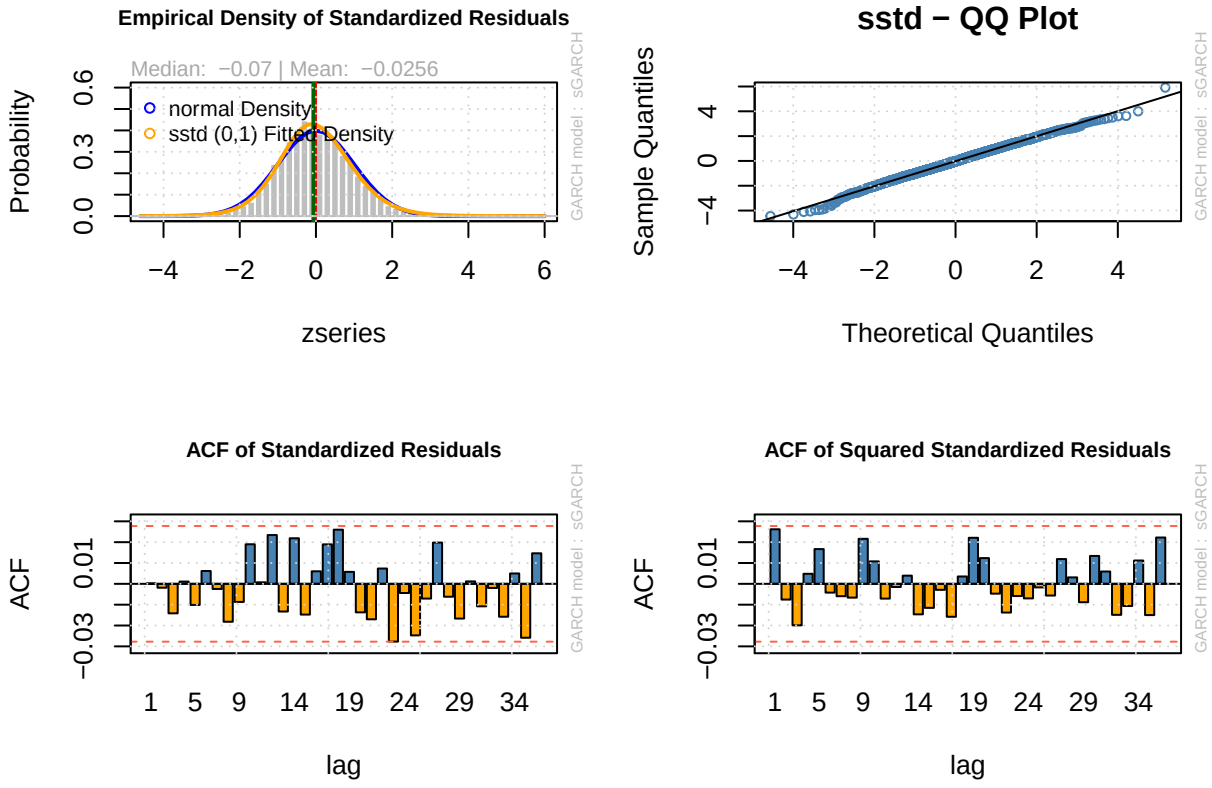
The **Jarque-Bera** statistic tests the residuals of the fit for normality based on the observed skewness and kurtosis, and **it appears that the residuals have non-normal skewness and kurtosis**.

The **Shapiro-Wilk** statistic tests the residuals of the fit for normality based on the empirical order statistics.

Both p-values of Jarque-Bera and Shapiro-Wilk tests allow us to reject the null hypothesis of normality assumption of the standardized residuals.

All **Ljung-Box Tests** are not significant together with LM ARCH test, which is also not significant. It means that the ARCH effects have been accommodated by the model and the chosen GARCH(1,1) model fits the series well.

Next, we look at 4 plots: Empirical Density of Standardized Residuals, QQ-Plot of Standardized Residuals, ACF of Standardized Residuals, ACF of Squared Standardized Residuals.



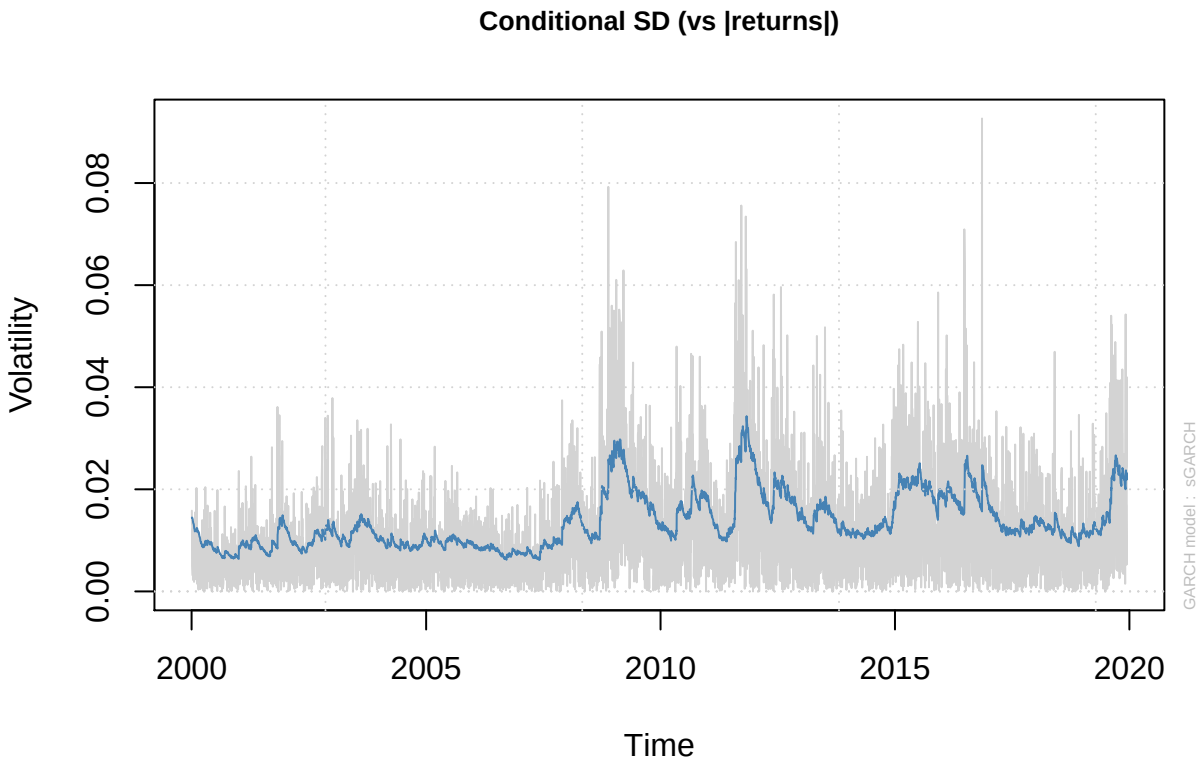
When look at the plot of Empirical Density of Standardized Residuals: we can see that the distribution is quiute close to normal.

When look at the QQ-Plot of Standardized Residuals: it is a quite straight line, because we alreaday chose the skew student-t distribution ("sstd").

When look at the plots of ACF of Standardized Residuals and ACF of Squared Standardized Residuals: they all show white noise pattern.

Thus, I feel the model seems perform quite well from the diagnostic checking.

The Fitted Conditional Variances Plot also shown below:



6 Volatility Forecasting

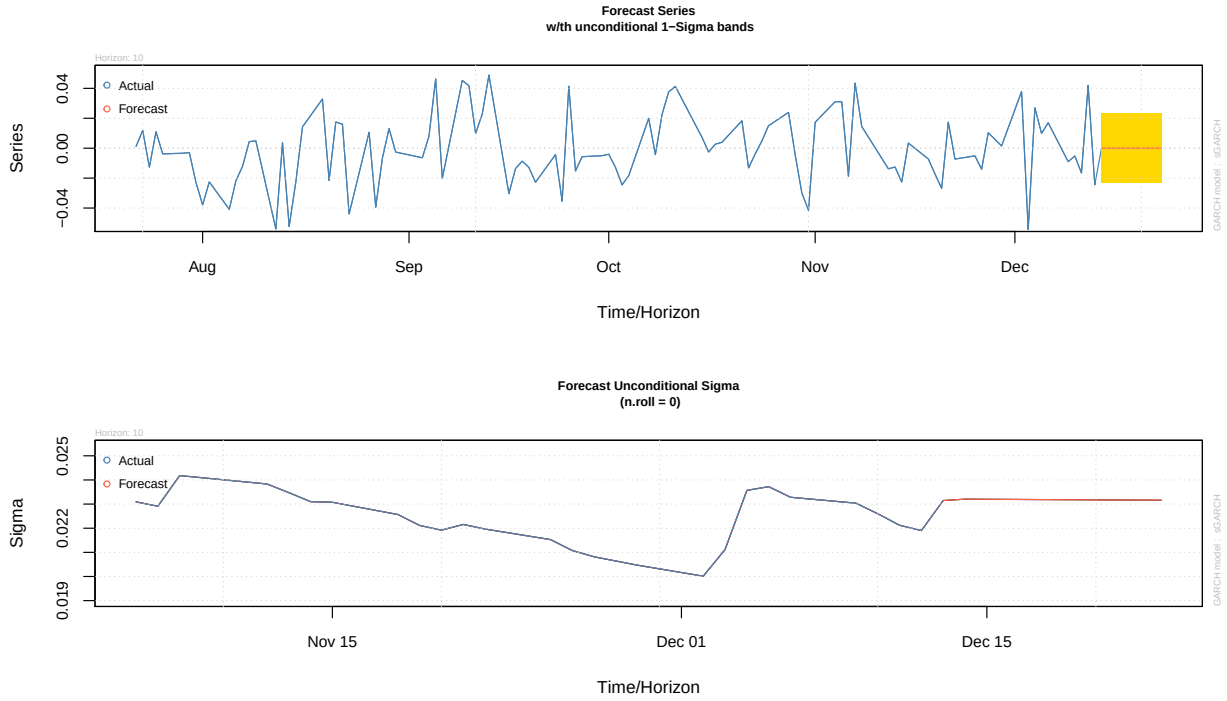
Using the rugarch package, we are able to forecast the unconditional out-of-sample volatility, but only for short periods of time, the longer the time horizon the more undependable the forecast is, so we attempt to forecast ten days ahead (2 weeks tradings on the market).

The results for 10 days predicted values of shown in below table:

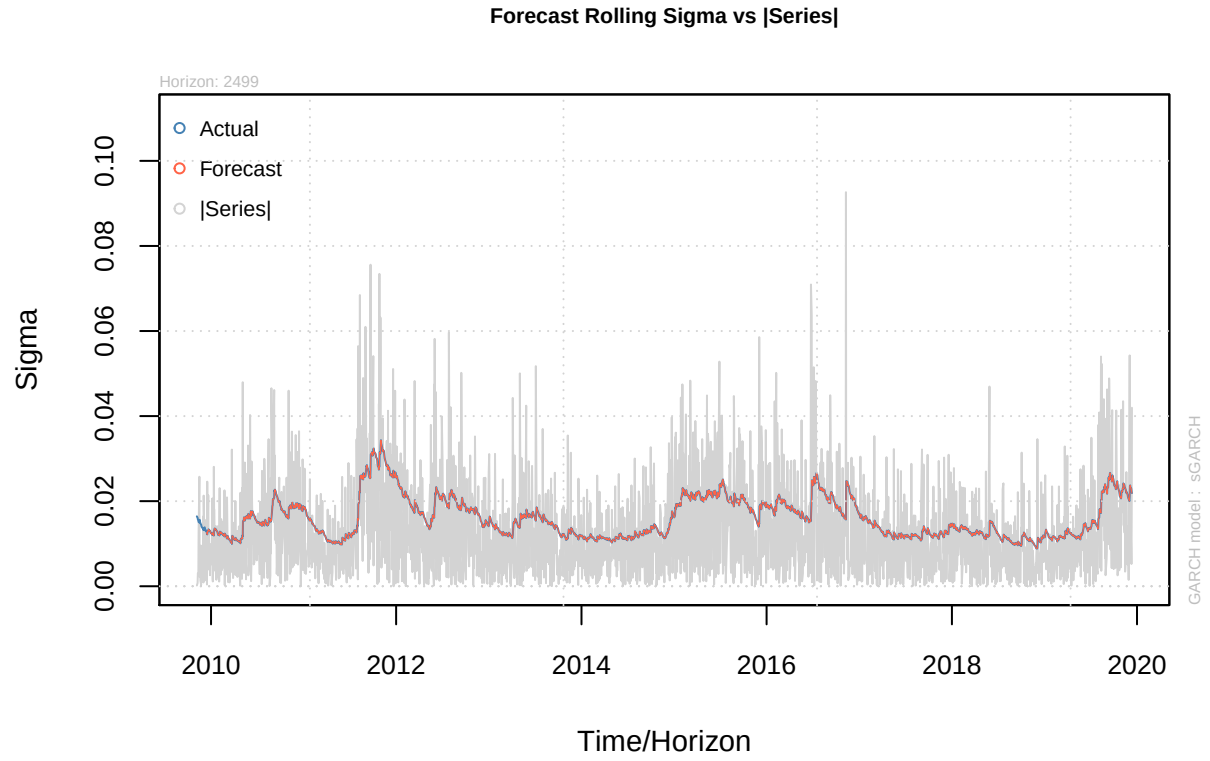
Table 8: 10-day prediction of volatility

	2019-12-13
T+1	0.0232039
T+2	0.0231990
T+3	0.0231941
T+4	0.0231893
T+5	0.0231844
T+6	0.0231796
T+7	0.0231747
T+8	0.0231699
T+9	0.0231650
T+10	0.0231602

We can plot the 10 days predictions as well:



We can also generate a rolling forecasts of σ_t plot them versus $|y_t|$.



It looks like to me this model does a pretty good job of sensing how long a volatility spike will remain elevated, or rather modeling the path of a volatility spike back down to long-run mean levels. Since all econometric models use past values to predict current values, it cannot foresee the initial spike up in volatility.

7 Conclusion

In this project, I used GARCH model to model and forecast the volatility of US bond yields. The whole dataset has historical data started since 1961 as present; there are 30 maturities from 1 year to 30 years. I only analyzed the bond with 20-year maturity and only looked at the recent years (since 2000).

The model I fitted seems performed pretty good, also it is simply a GARCH(1,1) model with skew student-t distribution ("sstd").

As I mentioned earlier in Exploratory Data Analysis, the yield curve tend to have a property that: the long yields (e.g. SVENY30) tend to be more stable in the long term, while the short yields (e.g. SVENY01) vary a lot. Also ACF and PACF plots further show this which implies that although 20-year maturity bond don't have ARMA model, while short yields bonds tend to have ARMA+GARCH model.

Actually, I did some futher analysis to test all 30 maturities of bonds. It seems that bonds with maturity term less than 10 year need to add the ARMA model, especially 1-year matrity bond seems need a quite complicated ARMA+GARCH model. Thus, for next step, I will further work on other maturity term bonds.

I will also try yield change (first order differencing) at the same time while I only used log return in this project (I plotted both yield change and log return plots at the beginning while only focused on log returns latter). This is because yield change is very commonly used in bond yield analysis together with yield returns, which is a bit different from stock price analysis. Two ways should be very similar, thus may provide us double confirmation.

8 R Code

```
knitr::opts_chunk$set(echo = TRUE)

# Load library
library(xts)
library(readr)
library(zoo)
library(viridis)
library(ggplot2)
library(ggfortify)
library(tseries)
library(TSA)
library(forecast)
library(fGarch)
library(rugarch)

# Load the data
yc_raw = read_csv("D:/d/Courses/STA/STA 9701/Project 2/FED-SVENY.csv")
# Convert the data into xts format
yc_all = as.xts(x = yc_raw[, -1], order.by = yc_raw$Date)

# Plot the time series
par(mfrow=c(1,1))
```

```r
plot.zoo(yc_all, ylab = "Yields", plot.type = "single", col = viridis(30),
         main="30 Maturities of US Government Bond Yields: 6/14/1961 - 12/13/2019")
# Add the legend
legend(x = "topleft", legend = colnames(yc_all),
       col = viridis(30), cex = 0.45, lwd = 3)

# Yield Change - differencing
# Differentiate the time series
yc_all_d = diff.xts(yc_all)
# Plot the differentiated time series
par(mfrow=c(1,1))
plot.zoo(yc_all_d, plot.type = "multiple",
         ylim = c(-0.5, 0.5), cex.axis = 0.7,
         ylab = 1:30, col = viridis(30),
         main="Daily Yield Changes of 30 Maturities of US Government Bond")

# daily log return
# log return of time series
yc_all_ld = diff.xts(log(yc_all))
# Plot the log return time series
par(mfrow=c(1,1))
plot.zoo(yc_all_ld, plot.type = "multiple",
         ylim = c(-0.4, 0.4), cex.axis = 0.7,
         ylab = 1:30, col = viridis(30),
         main="Log Returns of 30 Maturities of US Government Bond Yield")

# log return
yc_ld = yc_all_ld["2000/",]
# Save the 20-year maturity Log Returns into separate variable
x_ld_20 = yc_ld,"SVENY20"]
# plot log returns
par(mfrow=c(1,1))
autoplot(x_ld_20, main="Log Returns of 20-Year Maturity Bond Since 2000")

# adf test
adf.test(x_ld_20)

# Mean
mean(x_ld_20)
# t test, two-sided
t.test(x_ld_20)

par(mfrow=c(1,2))
# Histogram
hist(x_ld_20, xlab="Daily Log Returns", prob=TRUE,
     main="Histogram for Daily Log Returns")
xfit=seq(min(x_ld_20),max(x_ld_20),length=40)
yfit=dnorm(xfit,mean=mean(x_ld_20),sd=sd(x_ld_20))
lines(xfit, yfit, col=4, lwd=2)
# QQ-Plot
qqnorm(x_ld_20, main='Normal Q-Q Plot of Daily Log Returns'); qqline(x_ld_20, col=4)

# ACF and PACF of log returns, abs log returns, squared log returns
```

```r
par(mfrow=c(2,3))
acf_ld_20      =  acf(x_ld_20,main='ACF of log return data')
acf_abs_ld_20  =  acf(abs(x_ld_20),main='ACF of absolute log return data')
acf_sld_20     =  acf(x_ld_20^2,main='ACF of squared log return data')
pacf_ld_20     = pacf(x_ld_20,main='PACF of log return data')
pacf_abs_ld_20 = pacf(abs(x_ld_20),main='PACF of absolute log return data')
pacf_sld_20    = pacf(x_ld_20^2,main='PACF of squared log return data')

# McLeod-Li Test for Log Returns
par(mfrow=c(1,1))
McLeod.Li.test(y=x_ld_20, main='McLeod-Li Test for Log Returns of 20-Year Maturity Bond')

# Use auto.arima function
ARIMAfit_d_20 = auto.arima(x_ld_20, approximation=FALSE,trace=FALSE)
summary(ARIMAfit_d_20)

# fit the model and try AIC and BIC
g1.model = garchFit(~garch(1,0), x_ld_20, include.mean = F, trace=F)
(g1.model@fit)$ics
g2.model = garchFit(~garch(1,1), x_ld_20, include.mean = F, trace=F)
summary(g2.model)
(g2.model@fit)$ics
g3.model = garchFit(~garch(1,2), x_ld_20, include.mean = F, trace=F)
(g3.model@fit)$ics
g4.model = garchFit(~garch(2,1), x_ld_20, include.mean = F, trace=F)
(g4.model@fit)$ics
g5.model = garchFit(~garch(2,2), x_ld_20, include.mean = F, trace=F)
(g5.model@fit)$ics

# garchFit
# norm
g11_norm  = garchFit(~garch(1,1), x_ld_20, include.mean = F, trace=F,
                     cond.dist = "norm")
summary(g11_norm)
# snorm
g11_snorm = garchFit(~garch(1,1), x_ld_20, include.mean = F, trace=F,
                     cond.dist = "snorm")
summary(g11_snorm)
# std
g11_std   = garchFit(~garch(1,1), x_ld_20, include.mean = F, trace=F,
                     cond.dist = "std")
summary(g11_std)
# sstd
g11_sstd  = garchFit(~garch(1,1), x_ld_20, include.mean = F, trace=F,
                     cond.dist = "sstd")
summary(g11_sstd)

# rugarch
# norm
g_norm = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                    mean.model = list(armaOrder = c(0, 0), include.mean = F),
                    distribution.model = "norm")
fit_norm = ugarchfit(spec = g_norm, data = x_ld_20)
```

```
fit_norm
coef(fit_norm)
infocriteria(fit_norm)
gof(fit_norm,c(20,30,40,50))
plot(fit_norm, which='all')
# snorm
g_snorm = ugarchspec(variance.model = list(model = "sGARCH", garchOrder=c(1, 1)),
                     mean.model = list(armaOrder = c(0, 0), include.mean = F),
                     distribution.model = "snorm")
fit_snorm = ugarchfit(spec = g_snorm, data = x_ld_20)
fit_snorm
coef(fit_snorm)
infocriteria(fit_snorm)
gof(fit_snorm,c(20,30,40,50))
plot(fit_snorm, which='all')
# std
g_std = ugarchspec(variance.model = list(model = "sGARCH", garchOrder=c(1, 1)),
                     mean.model = list(armaOrder = c(0, 0), include.mean = F),
                     distribution.model = "std")
fit_std = ugarchfit(spec = g_std, data = x_ld_20)
fit_std
coef(fit_std)
infocriteria(fit_std)
gof(fit_std,c(20,30,40,50))
plot(fit_std, which='all')
# sstd
g_sstd = ugarchspec(variance.model = list(model = "sGARCH", garchOrder=c(1, 1)),
                     mean.model = list(armaOrder = c(0, 0), include.mean = F),
                     distribution.model = "sstd")
fit_sstd = ugarchfit(spec = g_sstd, data = x_ld_20)
fit_sstd
coef(fit_sstd)
infocriteria(fit_sstd)
gof(fit_sstd,c(20,30,40,50))
plot(fit_sstd, which='all')

# gof test table
knitr::kable(
  # norm
  gof(fit_norm,c(20,30,40,50)),
  caption = 'Goodness-of-Fit Test with "norm"'
)
knitr::kable(
  # snorm
  gof(fit_snorm,c(20,30,40,50)),
  caption = 'Goodness-of-Fit Test with "snorm"'
)
knitr::kable(
  # std
  gof(fit_std,c(20,30,40,50)),
  caption = 'Goodness-of-Fit Test with "std"'
)
knitr::kable(
```

```
  # sstd
  gof(fit_sstd,c(20,30,40,50)),
  caption = 'Goodness-of-Fit Test with "sstd"'
)

# robust coef matrix
knitr::kable(
  fit_sstd@fit$robust.matcoef,
  caption = 'Estimation Results (Robust Standard Error)'
)

# rugarch plot
par(mfrow=c(2,2))
plot(fit_sstd, which=8)
plot(fit_sstd, which=9)
plot(fit_sstd, which=10)
plot(fit_sstd, which=11)

# Fitted Conditional Variances Plot
plot(fit_sstd, which=3)

# 10 days forecasting
preds = ugarchforecast(fit_sstd, n.ahead=10)
show(preds)
sigma(preds)

# prediction table
knitr::kable(
  sigma(preds),
  caption = '10-day prediction of volatility'
)

# prediction plot
par(mfrow=c(2,1))
plot(preds, which = 1)
plot(preds, which = 3)

# rolling forecasting
spec           = getspec(fit_sstd)
setfixed(spec) = as.list(coef(fit_sstd))
garch.forecast = ugarchforecast(spec, n.ahead = 10, n.roll = 2499,
                                data = x_ld_20, out.sample = 2500)
show(garch.forecast)
sigma(garch.forecast)
plot(garch.forecast, which = 4)
```