ARMA-GARCH

SHIHENG SHEN

May 29, 2017

1 Data

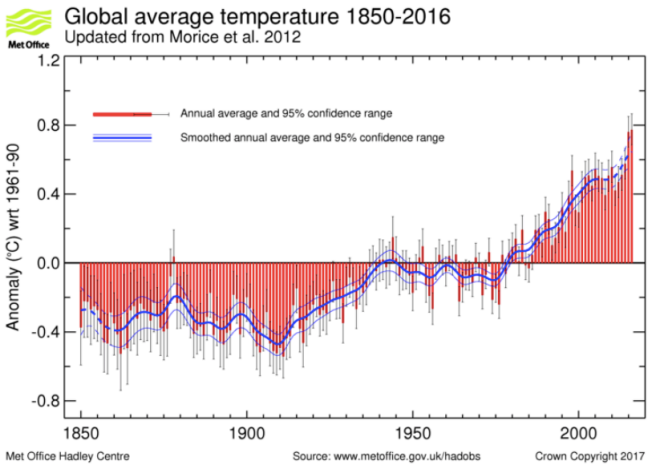
1.1 Data Source

HadCRUT4 is a gridded dataset of global historical surface temperature anomalies relative to a 1961-1990 reference period. Data are available for each month since January 1850, on a 5 degree grid. The dataset is a collaborative product of the Met Office Hadley Centre and the Climatic Research Unit at the University of East Anglia.
url =`http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/time_series/HadCRUT.4.5.0.0.monthly_ns_avg.txt`.

1.2 Brief Data Description

The gridded data are a blend of the CRUTEM4 land-surface air temperature dataset and the HadSST3 sea-surface temperature (SST) dataset. The dataset is presented as an ensemble of 100 dataset realisations that sample the distribution of uncertainty in the global temperature record given current understanding of non-climatic factors affecting near-surface temperature observations. This ensemble approach allows characterisation of spatially and temporally correlated uncertainty structure in the gridded data, for example arising from uncertainties in methods used to account for changes in SST measurement practices, homogenisation of land station records and the potential impacts of urbanisation. The HadCRUT4 data are neither interpolated nor variance adjusted.

Below is a graph provided by `www.metoffice.gov.uk`:



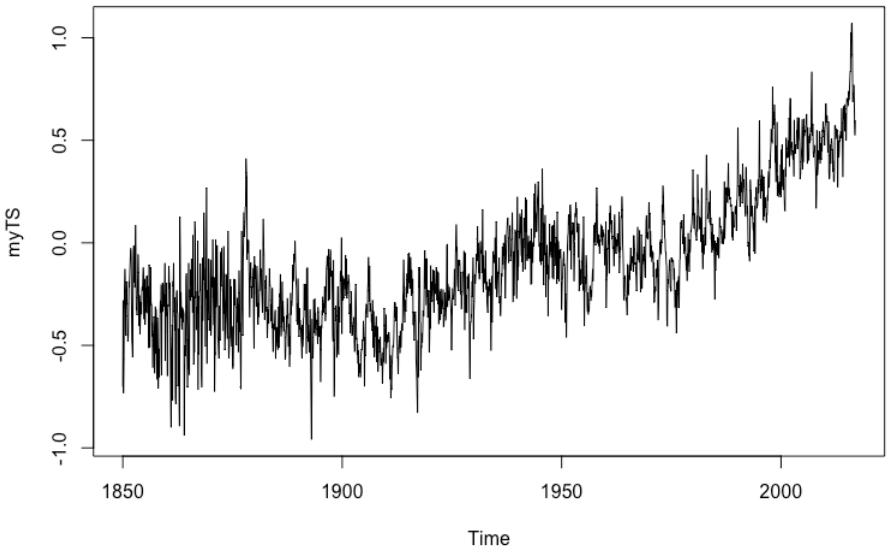
1.3 Loading the Data into R

For our purpose, we are not interested in the spatial distribution, so we only use the *Global Mean*.
Loading the data into R:

```
library(curl)
tmpf <- tempfile()
curl_download(url, tmpf)
gtemp <- read.table(tmpf)[, 1:2]
temp = gtemp$V2[1:2004]
```

We only use the first two columns: *time and monthly global mean*. We pick the first 2004 observations, which is monthly data from 1850 $\sim$ 2016, so that the data has complete periods.

The time series of *temp* is shown below:



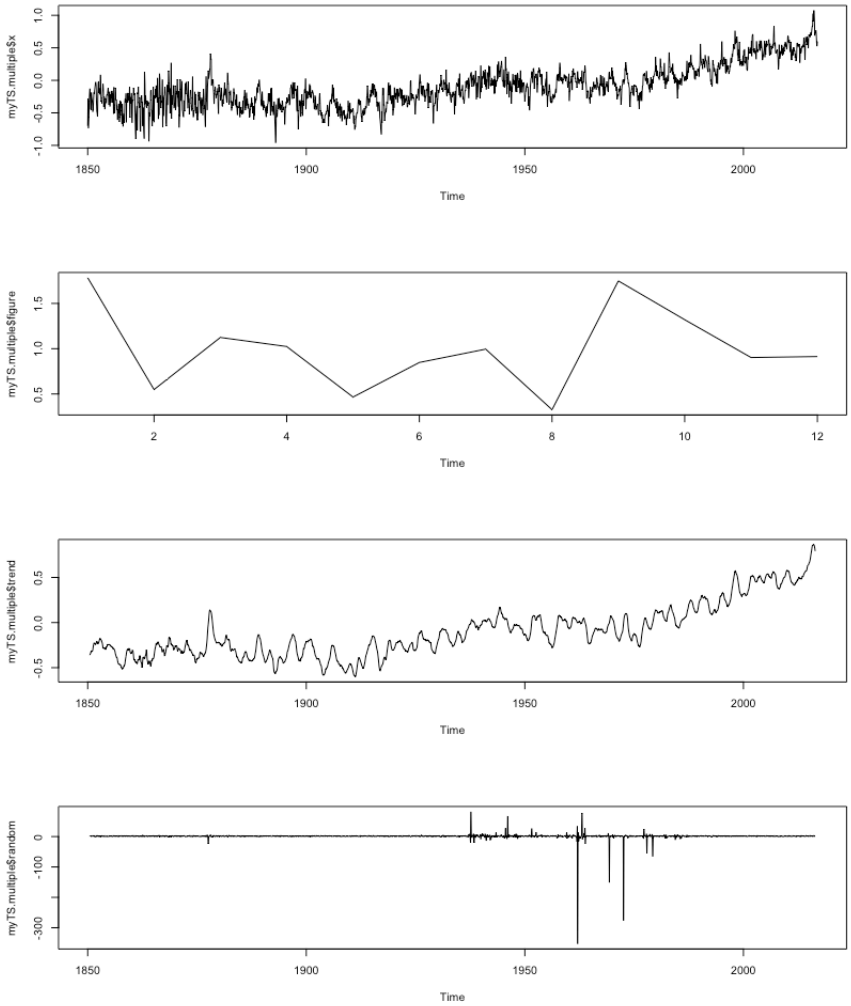
2 Dealing with the time series

2.1 Decompose

The first thing we might want to do is to wipe out the seasonal component and the time trend. There're many approaches to do it in R. We choose to use the *decompose*() provided by {*stats*}. Looking at the time series, we think a additive model should be appropriate.

We can quickly checked that multiplicative model is not appropriate:

Figure 1: multiplcative model: original ts, seasonal component, time trend, residuals



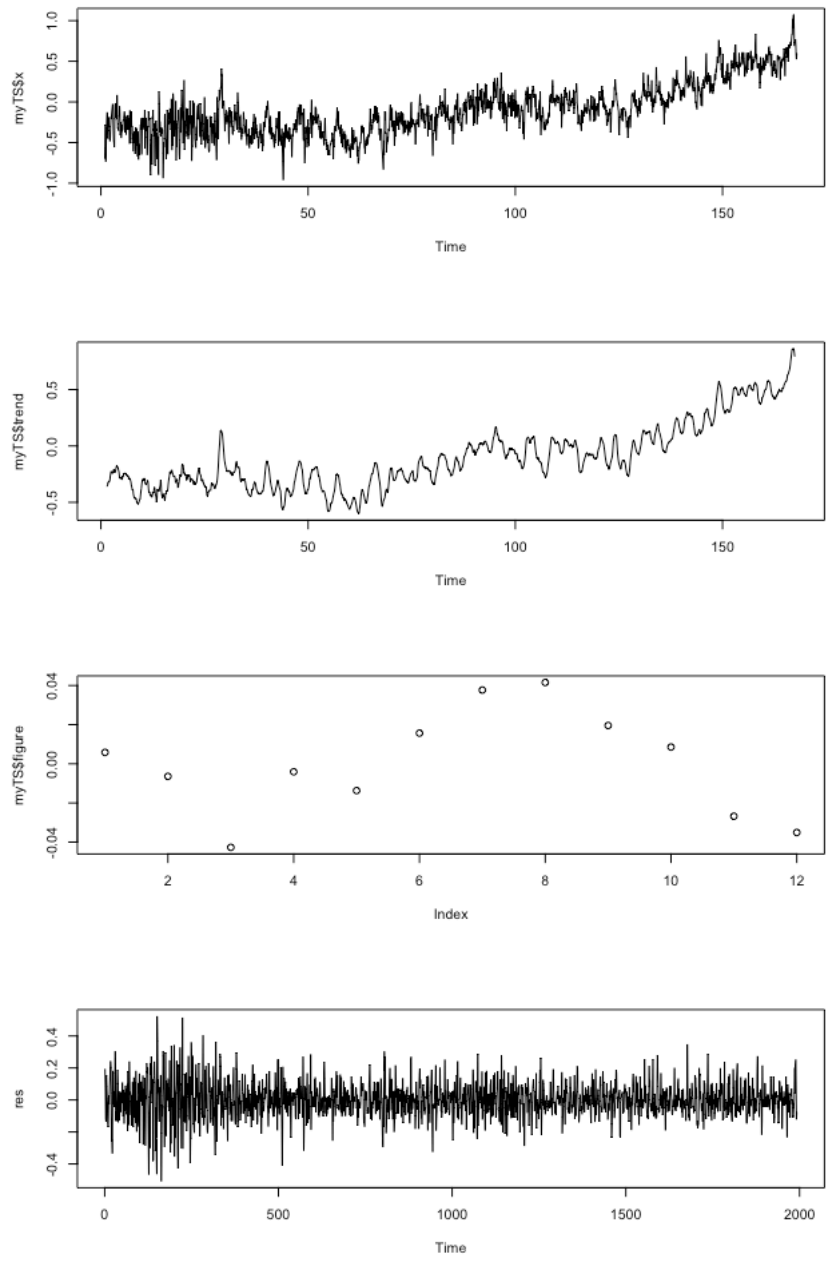
Obviously the random term is not easy to be modelled.

Therefore, decompose the time series using the following code:

```
library(TSA)
myTS = ts(as.numeric(temp), start = c(1850, 1), frequency = 12)
myTS.additive = decompose(myTS)
myTS.mutiple = decompose(myTS, type = "multiplicative")
# TS$x = TS$seasonal + TS$trend + TS$random(residuals)
res = myTS$random
# wipe out NAs
res.noNA = res[7:2004]
res = res.noNA[1:1992]
```

The *res* now is free of seasonal component and time trend. The following graph illustrate the components.

Figure 2: additive model: original ts, time trend, seasonal component, residuals



brief results:

```
> mean(res)
[1] 0.0002073084
> var(res)
[1] 0.01139539
> adf.test(res)
```

```
Augmented Dickey-Fuller Test

data:  res
Dickey-Fuller = -18.364, Lag order = 12, p-value = 0.01
alternative hypothesis: stationary
```

So we can convince ourself that the time series *res* is stationary.

2.2 Build a *ARMA(p, q)* for *res*

For simplicity, we use $auto.arima()$ from $\{forecast\}$.

```
library(forecast)
armamodel = auto.arima(res)
```

The results are all significant:

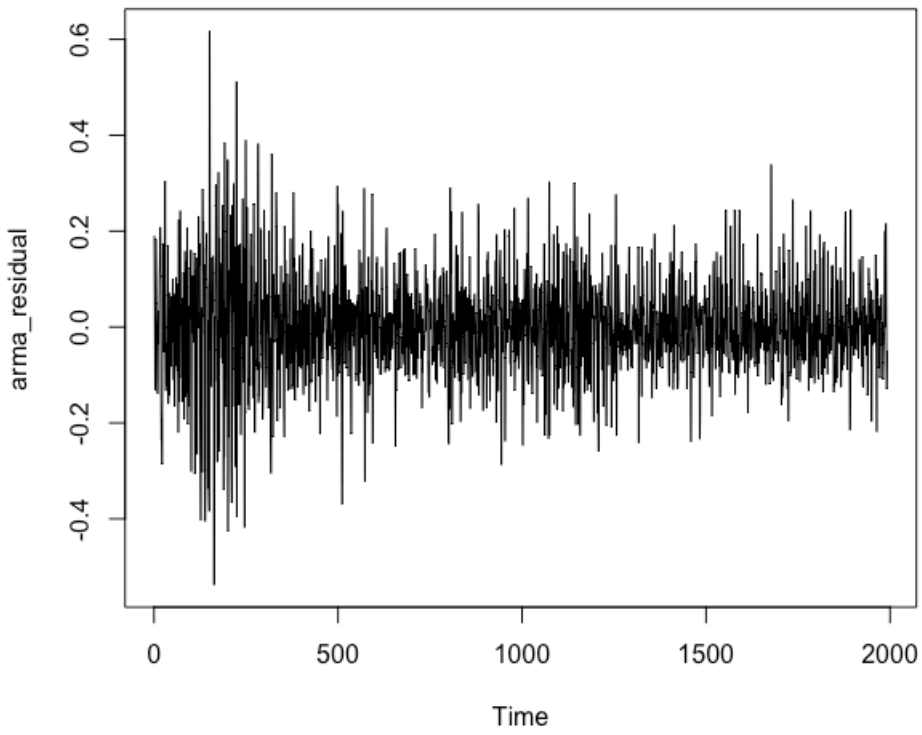
```
> armaModel
Series: res
ARIMA(1,0,2) with zero mean

Coefficients:
          ar1     ma1     ma2
      -0.5179  0.7198  0.1597
s.e.   0.1572  0.1550  0.0326

sigma^2 estimated as 0.01096:  log likelihood=1670.58
AIC=-3333.17   AICc=-3333.15   BIC=-3310.78
> Box.test(arma_residual, type = 'Ljung-Box')

Box-Ljung test

data:  arma_residual
X-squared = 0.086116, df = 1, p-value = 0.7692
```

The $ARMA(1, 2)$ success in making the arma residual independent.

2.3 GARCH

One can be pretty satisfied with the results right now. Yet Looking at the plot of arma residual:

Figure 3: arma residual



It makes us wondering if it has $ARCH$ effect. Using *arch.test* from $\{aTSA\}$ to perform the $LM - Test$:

```
library(aTSA)
> my.arma = arima(res, order = c(1, 0, 2))
> arch.test(my.arma)
ARCH heteroscedasticity test for residuals
alternative: heteroscedastic

Portmanteau-Q test:
     order   PQ p.value
[1,]     4  188       0
[2,]     8  233       0
[3,]    12  546       0
[4,]    16  665       0
[5,]    20  727       0
[6,]    24 1035       0
Lagrange-Multiplier test:
     order   LM  p.value
[1,]     4 1188 0.00e+00
```

```
[2,]     8  577 0.00e+00
[3,]    12  341 0.00e+00
[4,]    16  199 0.00e+00
[5,]    20  155 0.00e+00
[6,]    24  127 2.22e-16
```

It supports our doubt. Therefore we have the motive to build a $GARCH$ model. We decide to use the package *rugarch*.

Though we do know that using $GARCH$ may change the order of $ARMA$, yet given our significance of $ARMA$ model and the difficulty in determing the order, we decide to only model the variance in 'sGARCH'(standard GARCH model):

$$\sigma_t^2 = (w + \sum_{j=1}^{m} \zeta_j v_{jt}) + \sum_{j=1}^{q} \alpha_j \varepsilon_{t-j}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

So we wrote a function:

```
library(rugarch)
my_sGARCH_test <- function(p, q, m, n, ts.data = res)
{
# I use include.mean = FALSE after trying TRUE
# to find out insignificance
    myspec=ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(p, q)),
     mean.model = list(armaOrder = c(m, n), include.mean = FALSE),
     distribution.model = "norm")
    myfit=ugarchfit(myspec,data=ts.data, solver="solnp")
    return(myfit)
}
```

After trying a few times from $(1, 0)$ to $(5, 5)$, *GARCH(1 ,1)* is the most satisfying model.

```
> fit = my_sGARCH_test(1, 1, 1, 2, res)
> fit

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model : ARFIMA(1,0,2)
Distribution : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
ar1    -0.400525    0.187007  -2.1418 0.032213
```

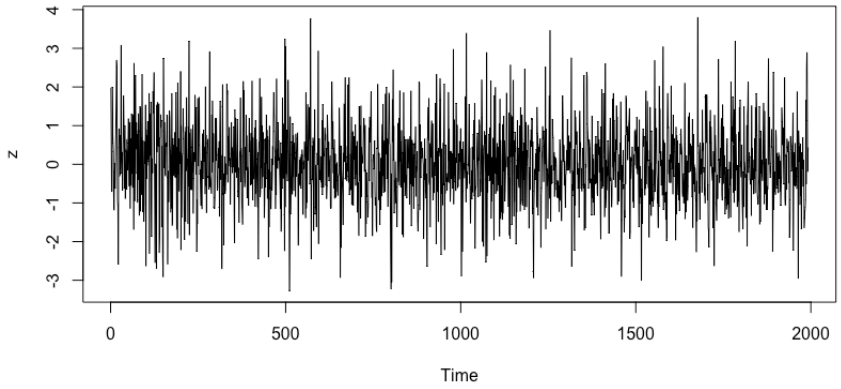
```
ma1     0.618016    0.184788   3.3445 0.000824
ma2     0.145898    0.039784   3.6672 0.000245
omega   0.000079    0.000022   3.5151 0.000440
alpha1  0.026802    0.003127   8.5718 0.000000
beta1   0.965023    0.002942 328.0210 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ar1    -0.400525    0.113813  -3.5191 0.000433
ma1     0.618016    0.109338   5.6524 0.000000
ma2     0.145898    0.029318   4.9764 0.000001
omega   0.000079    0.000024   3.3133 0.000922
alpha1  0.026802    0.003460   7.7452 0.000000
beta1   0.965023    0.001380 699.4205 0.000000

LogLikelihood : 1802.235

Information Criteria
------------------------------------

Akaike       -1.8034
Bayes        -1.7866
Shibata      -1.8035
Hannan-Quinn -1.7973

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic p-value
Lag[1]                     0.03983  0.8418
Lag[2*(p+q)+(p+q)-1][8]   83.58401  0.0000
Lag[4*(p+q)+(p+q)-1][14] 115.46992  0.0000
d.o.f=3
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      19.95 7.952e-06
Lag[2*(p+q)+(p+q)-1][5]     25.77 4.870e-07
Lag[4*(p+q)+(p+q)-1][9]     35.20 1.604e-08
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale  P-Value
ARCH Lag[3]   0.01004 0.500 2.000 0.920201
```

```
ARCH Lag[5]   4.34569 1.440 1.667 0.145206
ARCH Lag[7]  12.67065 2.315 1.543 0.004281

Nyblom stability test
------------------------------------
Joint Statistic:  1.8386
Individual Statistics:
ar1    0.04232
ma1    0.02495
ma2    0.31694
omega  0.09752
alpha1 0.49017
beta1  0.22730

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:       1.49 1.68 2.12
Individual Statistic:  0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value      prob sig
Sign Bias           0.4332 6.649e-01
Negative Sign Bias  3.7994 1.494e-04 ***
Positive Sign Bias  3.3241 9.031e-04 ***
Joint Effect       26.4729 7.593e-06 ***


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     64.71    6.813e-07
2    30     77.58    2.605e-06
3    40     90.01    6.563e-06
4    50    112.02    7.711e-07


Elapsed time : 0.2998619
```

Substract the standardized(w.r.t. the variance model) residuals z, which is $z = \frac{residuals(fit)}{sigma(fit)}$.

```
z = residuals(fit) / sigma(fit)
plot.ts(z)
```

Figure 4: z



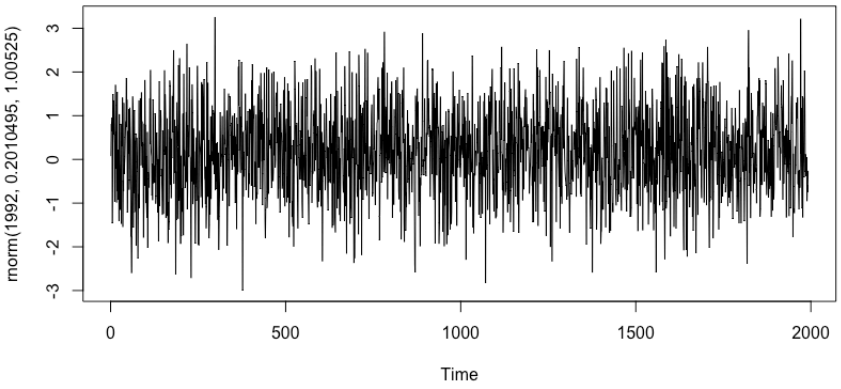
We can compute the mean and variance of z:

```
> mean(z)
[1] 0.02010495
> var(z)
[1] 1.00525
```

And then, just for fun, plot a normal sample series with the same parameters:

```
plot.ts(rnorm(1992, 0.2010495, 1.00525))
```

Figure 5: simulation of using rnorm



At least a human can't distinguish between them anymore. But what we most care about is whether standardized squared residuals can pass the $LM-Test$(substracting from the above long result):

```
Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale  P-Value
ARCH Lag[3]   0.01004 0.500 2.000 0.920201
ARCH Lag[5]   4.34569 1.440 1.667 0.145206
ARCH Lag[7]  12.67065 2.315 1.543 0.004281
```

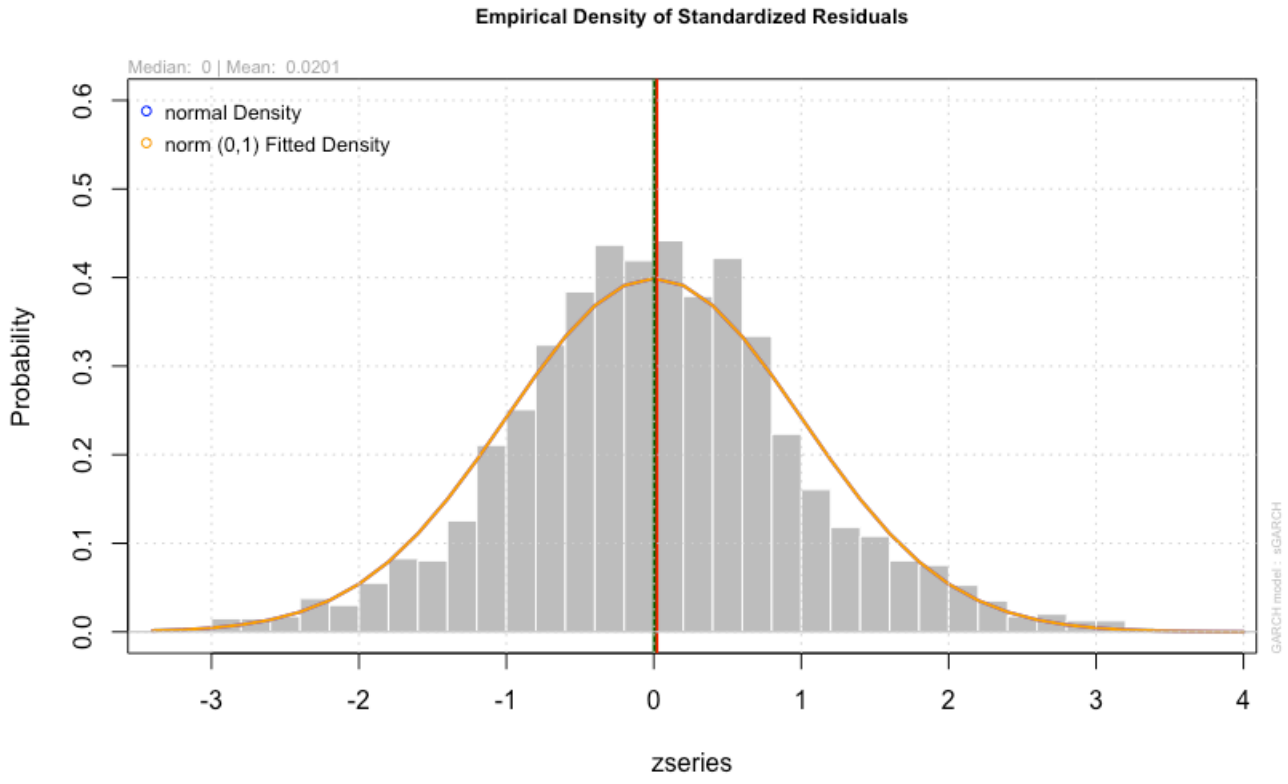
We can see our model successfully wipes out ARCH effect at smaller lags, but fails at $lag[7]$. This is the best 'sGARCH' can give with respest to our data.

Graphical Diagnostics:

rugarch provide a function to plot a "uGARCHfit" obeject.

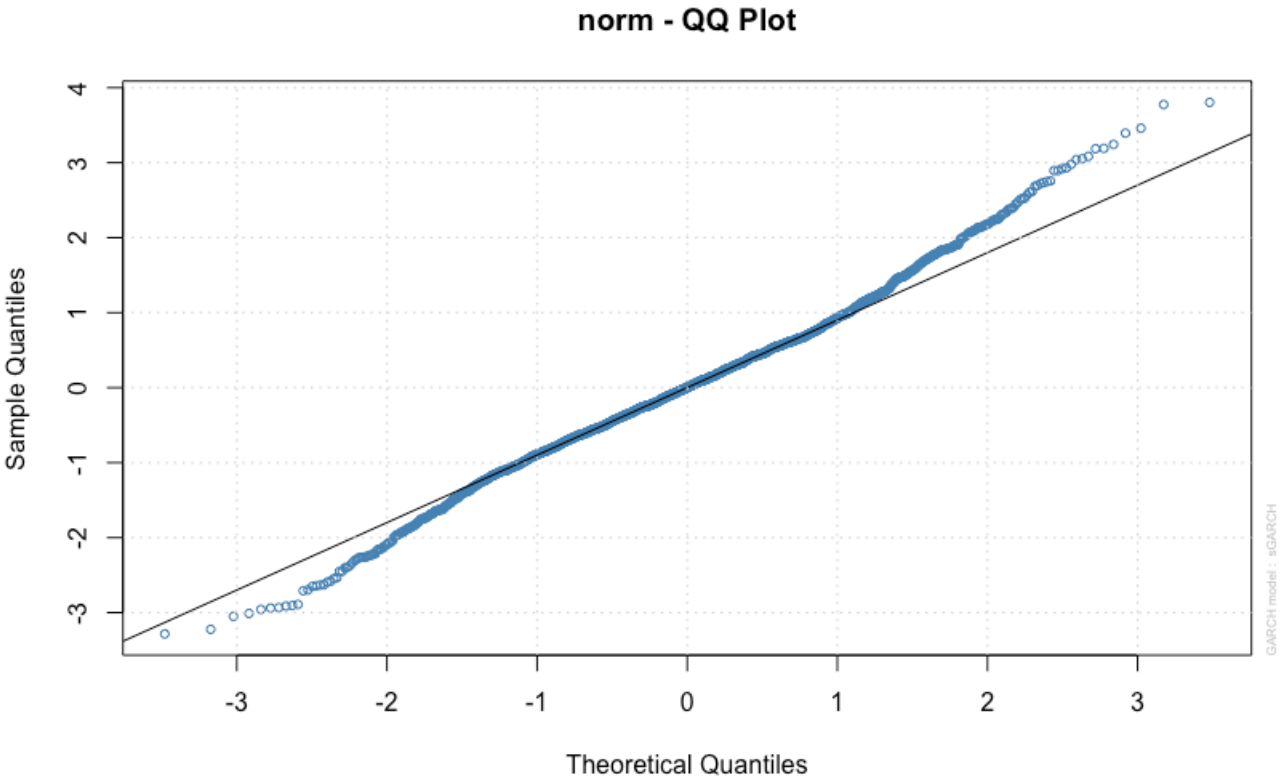
ex. density smmothed to normal distribution:

Figure 6: Empirical Density of Standardized Residuals



qqplot:

Figure 7: qqplot of z



We can see that z is skewed, which might be correlated with long lags' heterodasticity.

Conclusion: 'sGARCH' have improved the model in the intuitive sense and to some extent fix the heterodasticity.