ST436

23074

June 03, 2021

Contents

1 Q1

1.1 a

This paper raised a challenge against the Gaussian maximum-likelihood estimator (GMLE) that the estimator may not be asymptotically normal if the innovation distribution is heavy tailed with an inftyite fourth moment, causing the convergence rate slower than the standard rate. In response to this, the paper is suggesting a log-transform-based least absolute deviation estimator (LADE) as an alternative approach which is robust to the heavy tails of the innovation distribution. And propose a goodness-of-fit statistics to help select which distribution to choose. (GMLE if innovation distribution is a normal distribution, LADE if the log squared innovations is close to a Laplace distribution)

1.2 b

Re-writing the equation (2.1) in the paper using the notations from ST436 lecture notes:

$$X_t = \sigma_t \epsilon_t$$

$$\sigma_t^2 \equiv \sigma_t(\theta)^2 = a_0 + \sum_{i=1}^{p} a_i X_{t-i}^2 + \sum_{j=1}^{q} b_j \sigma_{t-j}^2$$

where $a_0, b_j, a_j \geq 0$, $\theta = (a_0, a_1, ..., a_p, b_1, ...b_q)^T$ and $\{\epsilon_t\}$ is a sequence of IID random variable with mean 0 and variance 1 and ϵ_t is an unknown distribution, independent of $\{X_{t-k}, k \geq 1\}$ for all t.

The necessary and sufficient condition for the above equation to be stationary process with $EX_t^2 < \infty$ is:

$$\sum_{i=1}^{p} a_i + \sum_{j=1}^{q} b_j < 1$$

Under this condition, $\sigma_t^2 = \sigma_t(\theta)^2$ can be shown as:

$$\sigma_t^2 = \frac{a_0}{1 - \sum_{j=1}^{q} b_j} + \sum_{i=1}^{\min(p,t-1)} a_i X_{t-i}^2 + \sum_{i=1}^{p} a_i \sum_{k=1}^{\infty} + \sum_{j_1=1}^{q} ... + \sum_{j_k=1}^{q} b_{j_1} ... b_{j_k} X_{t-j0j_1-...-j_k}^2$$

And GMLE of GARCH(p,q) in the paper can be re-written as:

$$f(X_1, ..., X_t|a) = \prod_{t=\max(p,q)+1}^{n} \frac{1}{(2\pi\sigma_t^2)^{1/2}} \exp(\frac{-X_t^2}{2\sigma_t^2}) \times f(X_1, ..., X_{\max(p,q)}|a)$$

where $a = (a_0, a_1, ..., a_{\max(p,q)})$

Then the target function for maximisation is:

$$f(X_{\max(p,q)+1}, ..., X_n | a, X_1, ..., X_{\max(p,q)}) = \prod_{t=\max(p,q)+1}^{n} \frac{1}{(2\pi\sigma_t^2)^{1/2}} \exp(\frac{-X_t^2}{2\sigma_t^2})$$

Applying the logs then multiplication of -1 leads to the minimisation of the function w.r.t. $\hat{\theta}$:

$$\hat{\theta} = \arg\min_{\theta} \sum_{t=v+1}^{T} [\log \tilde{\sigma}_t^2 + \frac{X^2}{\tilde{\sigma}_t^2}]$$

Where $\tilde{\sigma}_t^2$ is a truncated version of σ_t^2 and $v \geq 1$ is an integer that controls the effect of this truncation. For ARCH(P) process where $q = 0$, then we choose $v = p$.

$$\tilde{\sigma}_t^2 = \frac{a_0}{1 - \sum_{j=1}^{q} b_j} + \sum_{i=1}^{\min(p,t-1)} a_i X_{t-i}^2 + \sum_{i=1}^{p} a_i \sum_{k=1}^{\infty} \sum_{j_1=1}^{q} ... \sum_{j_k=1}^{q} a_{j_1} ... a_{j_k} X_{t-j0j_1-...-j_k}^2 I(t - i - j_1 - ... - j_k \geq 1)$$

In addition, the method can be further elaborated with the temporary assumption that $\epsilon_t \sim N(0,1)$. Given $\{X_k, k \leq v\}$ with $v \geq max(p,q)$, the conditional density function of $X_{v+1}, ..., X_n$ is then proportional to maximisation of the function below:

$$\max_{\theta} [\{ \prod_{t=v+1}^{n} \sigma_2^t \}^{-1/2} \exp\{-\frac{1}{2} \sum_{t=v+1}^{n} \frac{X_t^2}{\sigma_t^2}\}]$$

Comparing to the equations in ST436 lecture notes, the paper imposes truncated MLE with truncated $\tilde{\sigma}_t^2$. Apart from that, given the lecture note proposes the MLE of GARCH model with Gaussian innovations which follow the same logic as in the paper (same σ_t^2 structure).

1.3 c

Let $C_0 > 0$ be a constant that $median(e_t^2) = 1$ where $e_t = C_o^{1/2} \epsilon_t$ and naturally $median(log(e_t^2)) = 0$

$$s_t^2 \equiv s_t(\alpha)^2 = \gamma + \sum_{i=1}^{p} \beta_i X_{t-i}^2 + \sum_{j=1}^{q} b_j s_{t-j}^2$$

where: $s_t^2 = \sigma_t^2 / C_0, \gamma = a_o / C_0, \beta_i = a_i / C_0$ and $\alpha = (\gamma, \beta_1, ..., \beta_p, b_1, ...b_q)^T$.

Then the taking the log of X_t^2 becomes:

$$\log(X_t^2) = \log\{s_t(\alpha)^2\} + \log(e_t^2)$$

Based on the setting, the median of $log(e_t^2)$ becomes 0. Hence, the parameter α that we want to optimise with the minimisation of $E|\log(X_t^2) - \log\{s_t(\alpha)^2\}|$ motivates the LADE. The parameter to optimise for LADE is now $\hat{\alpha}$, and the minimisation function now becomes:

$$\hat{\alpha} = \arg\min_{\theta} \sum_{t=v+1}^{T} |\log(X_t^2) - \log(\tilde{s}_t(\alpha)^2)|$$

where $\tilde{s}_t(\alpha)^2$ is a truncated version of $s_t(\alpha)^2$:

$$\tilde{s}_t(\alpha)^2 = \frac{\gamma}{1 - \sum_{j=1}^{q} b_j} + \sum_{i=1}^{min(p,t-1)} \beta_{t-i} X_{t-i}^2 \sum_{i=1}^{p} \beta_i \sum_{k=1}^{\infty} \sum_{j_1=1}^{q} ... \sum_{j_k=1}^{q} b_{j_1} ... b_{j_k} X_{t-j0j_1-...-j_k}^2 I(t-i-j_1-...-j_k \geq 1)$$

which is directly linked to the $\tilde{\sigma}_t^2$ function in GMLE case but with different parameter optimisation.

The above minimisation function can also be re-written as a maximum quasilikelihood estimator with assumption that $\log(e_t^2)$ follows a Laplace distribution with density to $0.5\lambda \exp(-\lambda|x|)$, where constant $\lambda > 0$. Given $\{X_k, k \leq v\}$ with $v \geq max(p,q)$, the conditional density function of $X_{v+1}, ..., X_n$ is then proportional to maximisation of the function below:

$$\max_{a} \exp(-\lambda \sum_{t=v+1}^{T} |\log(X_t^2) - \log(s_t(\alpha)^2)|)$$

1.4 d

```
simulate_garch = function(params, n, d)  {
  #' GARCH(1,1) simulation function for given distribution.
  #' @param params vector. The sequence of parameters, a0, a1, b1 respectively
  #' @param n int. # of simulation size
  #' @param d vector. Sample population from the designated distribution
  a0 = params[1]
  a1 = params[2]
  b1 = params[3]
  sigma = a0 / (1 - a1 - b1)
  x0 = sqrt(sigma) * d[1]
  x = c(x0)
  for (i in 2:n) {
    sigma_new = a0 + a1 * x0^2 + b1 * sigma
    x1 = sqrt(sigma_new) * d[i]
    x = c(x, x1)
    x0 = x1
    sigma = sigma_new
  }
  return(x)
}

# Simulate GARCH (1,1) Series with normal distribution
x = simulate_garch(c(0.1, 0.6, 0.3) , 100, rnorm(100))
```

In order to test the GMLE function, GARCH(1,1) process with $a_0 = 0.5$, $a_1 = 0.6$ and $b_1 = 0.3$ is simulated for $n = 10000$ size using the custom GARCH function, simulate_garch.

```
set.seed(1)
gmle = function(x, params) {
  #' GMLE function in the paper.
  #' @param x vector. An array to be estimated
  #' @param params vector. The sequence of parameters, a0, a1, b1 respectively
  a0 = params[1]
  a1 = params[2]
  b1 = params[3]
  n = length(x)
  t_sig = c(a0/(1-b1) + a1 * x[1]^2) # Equation (2.5)
  t_sig = c(t_sig, a0 / (1 - b1) + a1 * x[2:(n-1)]^2 + a1 * cumsum(x^2*b1^c(n:1))[1:(n-2)])
  ll = sum(x[2:n]^2/t_sig + log(t_sig)) # Equation (2.4) in the paper
  if (a0>=0 && a1 >= 0 && b1 >= 0 && (a1+b1) < 1) { # Required constraint for GARCH equation
    return(ll)
  } else {return(1e+09)}}

garch_gmle <- function(x) {
  #' Optimisation for GMLE.
  #' @param x vector. An array to be estimated
  params = c(0.1, 0.1, 0.1) # initial parameters for optimisation
  result = optim(params, gmle, x=x, method = c("BFGS")) # BFGS Algorithm for minimisation
  res = data.frame(result$par)
  row.names(res) = c("alpha0","alpha1","beta")
  res["-LL", ] = result$value
  return(res)}
kable(garch_gmle(x), caption="GARCH MLE")
```

Table 1: GARCH MLE

	result.par
alpha0	0.9216021
alpha1	0.8177468
beta	0.0000347
-LL	120.8646943

Based on the result above, the custom GMLE function works as intended but the accuracy of the estimator needs to be tested and this will be further discussed in the next part (e).

```
lade_mle = function(x, params) {
  #' LADE function in the paper.
  #' @param x vector. An array to be estimated
  #' @param params vector. The sequence of parameters, a0, a1, b1, c0 respectively
  a0 = params[1] # c in the paper
```

```
  a1 = params[2] # b1 in the paper
  b1 = params[3] # a1 in the paper
  c0 = params[4] # C0 in the paper
  gamma = a0 / c0
  beta = a1 / c0
  n = length(x)
  s2 = c(gamma/(1-b1) + beta * x[1]^2) # Equation (2.6)
  s2 = c(s2, gamma / (1 - b1) + beta * x[2:(n-1)]^2 + beta * cumsum(x^2*b1^c(n:1))[1:(n-2)])
  ll = sum(abs(log(x[2:n]^2) - log(s2)))
  if (a0>=0 && a1 >= 0 && b1 >= 0 && a1+b1 < 1 && c0 > 0) { # Constraint for GARCH
    return(ll)
  } else {
    return(1e+09)}}

garch_lade <- function(x) {
  #' Optimisation for LADE.
  #' @param x vector. An array to be estimated
  params = c(0.1, 0.1, 0.1, 0.1) # initial parameters for optimisation
  result = optim(params, lade_mle, x=x, gr=NULL, method = c("L-BFGS-B"))
  res = data.frame(result$par)
  row.names(res) = c("a0","a1","b1", "C0")
  res["-LL", ] = result$value
  return(res)}
```

To check whether LADE function operates, the Laplace random variable generator function laplce_rv is defined.

```
laplce_rv <- function(n, mu=0, b=1) {
  #' Laplace r.v. generator, source: Wikipedia
  #' @param n int. A number of samples
  #' @param mu float. Location parameter
  #' @param lam float. Scale parameter
  p = runif(n)
  x1 = b * log(2 * p) + mu
  x2 = -b * log(2 * (1-p)) + mu
  x1[x1>mu] = 0
  x2[x2<=mu] = 0
  return(x1 + x2)}
# Simulate GARCH (1,1) Series with normal distribution
lp_x = simulate_garch(c(0.6, 0.7, 0.2), 100, laplce_rv(100))
kable(garch_lade(lp_x), caption="GARCH LADE")
```

Table 2: GARCH LADE

	result.par
a0	0.1046590
a1	0.0216791
b1	0.0957210
C0	0.2231011
-LL	177.3914947

Based on the result above, the defined LADE function works as intended but the accuracy of the estimator needs to be tested and this will be further discussed in the next part (e).

1.5 e

The paper illustrates that LADE function works better when the sample distribution is closer to Laplace distribution. To evaluate this, GARCH simulations with different innovation distributions are examined to validate the legitimacy of the statement from the paper. In line with the paper, 6 different GARCH(1,1) of ϵ_t distribution will be assessed:

1. N(0,1)
2. t with DoF=3
3. Skewed t with DoF=6
4. Skewed t with DoF=3
5. Laplace with mu=0 and b=1
6. Laplace with log

```
skewed_t_rv <- function(n, dof) {
  #' Skewed t r.v. generator
  #' @param n int. A number of samples
  #' @param dof int. Degrees of freedom
  v0 = abs(rnorm(n))
  v1 = rnorm(n)
  v2 = rchisq(n, dof)
  (0.8 * v0 + 0.6 * v1) / sqrt(v2 / dof)
}
```

```
n = 1000
cases = list(rnorm(n), rt(n, 3), skewed_t_rv(n, 6), skewed_t_rv(n, 3), laplce_rv(n, mu=0, b=1),
names(cases) = c("N(0,1)", "t with DoF=3", "Skewed t with DoF=6", "Skewed t with DoF=3", "Laplac
```

Skewed t and Laplace distributions are needed. Laplace r.v. generator function is shown in (d). For skewed t distribution, we are going to implement the same function as it is. The simulation is run for 200 replications as in the paper. And the paper utilises the estimation errors:

$$\epsilon_{Estimator} = |\hat{\alpha}_1/\hat{\alpha}_0 - \alpha_1/\alpha_0| + |\hat{b}_1 - b_1|$$

```r
ee_func = function(param_hat, param_r) {
  #' Estimation error function.
  #' @param param_hat vector. A vector of estimated parameters
  #' @param param_r vector. A vector of true parameters
  return(abs(param_hat[2]/param_hat[1]-param_r[2]/param_r[1]) + abs(param_hat[3]-param_r[3]))}
```

```r
params = c(0.5, 0.6, 0.3)
res = matrix(nrow=length(cases), ncol=2)
for (case_n in 1:length(cases)){
  d = cases[[case_n]] # Each case distribution
  e_gmle = c()
  e_lade = c()
  for (i in 1:5) {
    x = simulate_garch(params, n, d)
    gmle_res = garch_gmle(x)
    lade_res = garch_lade(x)
    eps_gmle = ee_func(as.vector(gmle_res[,1]), params)
    eps_lade = ee_func(as.vector(lade_res[,1]), params)
    e_gmle = c(e_gmle, eps_gmle)
    e_lade = c(e_lade, eps_lade)
  }
  res[case_n, ] = c(mean(e_gmle), mean(e_lade))
}
res = data.frame(res)
row.names(res) = names(cases)
colnames(res) = c("Epsilon_GMLE", "Epsilon_LADE")
kable(res, caption="Estimation Error by Distribution")
```

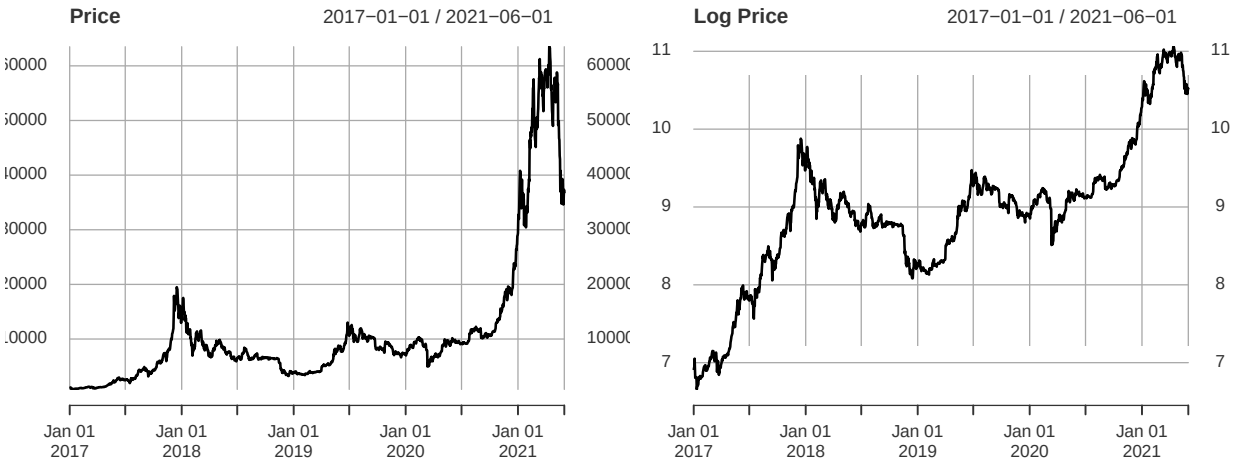
Table 3: Estimation Error by Distribution

	Epsilon_GMLE	Epsilon_LADE
N(0,1)	0.4344818	0.9231015
t with DoF=3	3.1802977	0.7636113
Skewed t with DoF=6	0.6013738	0.8903284
Skewed t with DoF=3	1.5905905	0.6012284
Laplace with mu=0 and b=1	1.8208587	0.8579272
Laplace with log	1.2568542	0.7124185

Based on the simulated results, the estimation error of GMLE is lower for innovations with N(0,1) and t(3). Meanwhile, LADE shows lower error for the non-normal distributions, suggesting that LADE is preferred method for GARCH series estimation for Laplace distribution. To conclude, this statistical result is consistent with the findings in the paper.

2 Q2

In this report, I have used time series analysis to introduce a model that explains and predicts the market behaviour of Bitcoin, the longest running and most well known cryptocurrency.



The time series plots of both volume and price of BTC indicate the dramatic trend changes as well as the extreme volatility clustering aspects in the recent observations, which reflect the well-known volatility & instability of BTC compared to other main-stream financial products such as stocks or bonds.

As the main focus of this report is to model and capture the rapid trend changes of BTC, the stable periods in the first 2 years are going to be excluded and data after January 2017 where BTC reached USD 1,000 mark for the first time in 3 years is going to be examined.

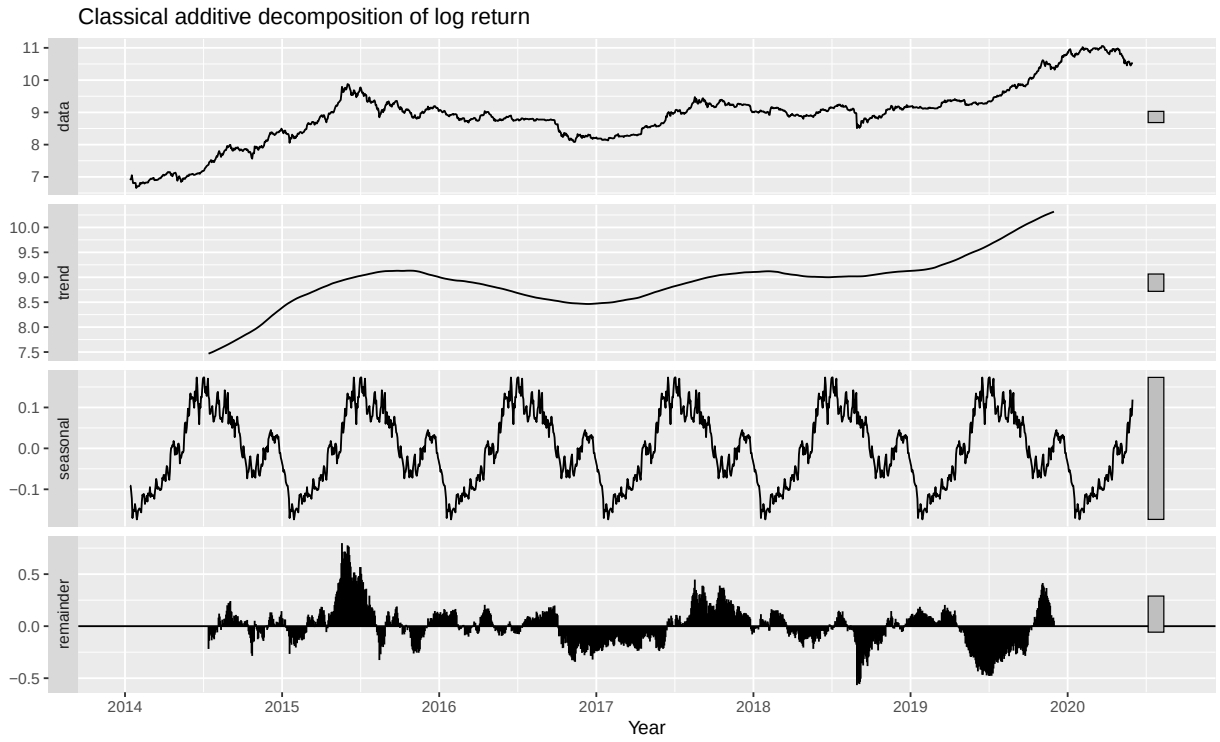
Compared to the original price time series (left), the log-transformed price series shows a significant improvement in terms of the lower variance of time series with mitigated dramatic trend change. In addition, log transformation prevents us from violating the statistical assumption that the product of normally distributed variables is not normal as the sum of normally distributed variables (log-series) is still normal. Hence, log-transformed price series is going to be examined.

Table 4: Summary Statistics: BTC Price

	BTC-USD.Adjusted
Mean	11300.00
Median	7950.00
SD	12800.00
Min	778.00
Max	63500.00
Skew	2.55
Kurt	8.83
#Obs	1610.00

Above summary statistics of BTC adjusted close price illustrates that there are 1600+ observations

(nearly 5 years) with a huge variance as well as a massive range between minimum and maximum.




```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 8 lags.
##
## Value of test-statistic is: 10.1533
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739


##
##  Augmented Dickey-Fuller Test
##
## data:  ts_v
## Dickey-Fuller = -2.1516, Lag order = 12, p-value = 0.5141
## alternative hypothesis: stationary
```

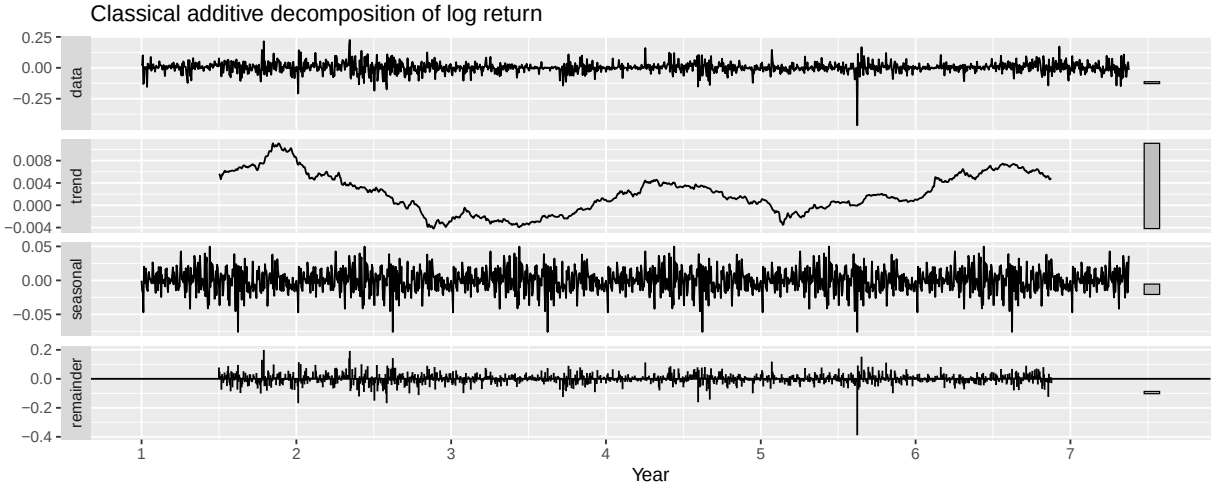
Here, the time series can be decomposed into three components, trend cycle (T_t), seasonal component (S_t) and random component (E_t), and can be represented as following formula:

$$Y_t = T_t + S_t + R_t.$$

With this additive decomposition, it is possible to separate the random component of the series from other two determinant series.

In addition, the Augmented Dickey-Fuller test and KPSS test statistics attest that the log price series is non-stationary. KPSS test statistic is much greater than p-value at 1%, meaning that there is a strong evidence of non-stationarity and ADF test statistic cannot reject the null hypothesis of non-stationarity as well. This indicates that the stationary time series is required. Differencing the integrated series could help stabilise the mean of a time series by removing trend and seasonality. Hence, the log return series is going to be examined. Following formula represents the log return series:

$$\log(return_t) = \log(price_t) - \log(price_{t-1}) = \Delta \log(price_t) \sim I(0)(Expected, Ideal)$$




```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 8 lags.
##
## Value of test-statistic is: 0.1476
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739


##
##  Augmented Dickey-Fuller Test
##
## data:  ts_v
## Dickey-Fuller = -10.432, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

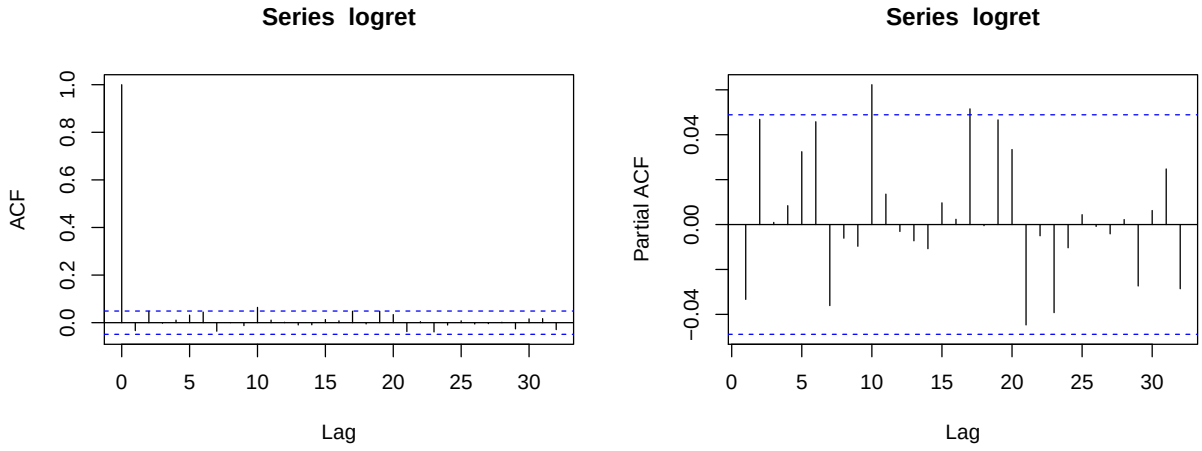
Time series plot of log return seems to be very close to a stationary process without trend nor a sign of increasing variance. This indicates that the first differencing has been successful at stabilising the integrated series. ADF test and KPSS test statistics further attest the stationarity. KPSS test statistic is much lower than p-value at 10%, meaning that there is a very weak evidence of non-stationarity and ADF test statistic rejects the null hypothesis of non-stationarity at 1% as well.

```
##
##  Box-Ljung test
##
## data:  logret
## X-squared = 1.7894, df = 1, p-value = 0.181
```

According to the Ljung-Box test, p-value does not reject the null hypothesis, indicates that the log return series is independently distributed, allowing this study to continue with a solid motivation.

2.1 ARMA Approach

The earlier efforts of securing the stationary property of time series comes from the basic setting that ARMA models require stationary data. In order to choose the right p and q terms of ARMA(p,q) model, the identification of the model using ACF & PACF (Correlogram) is needed.

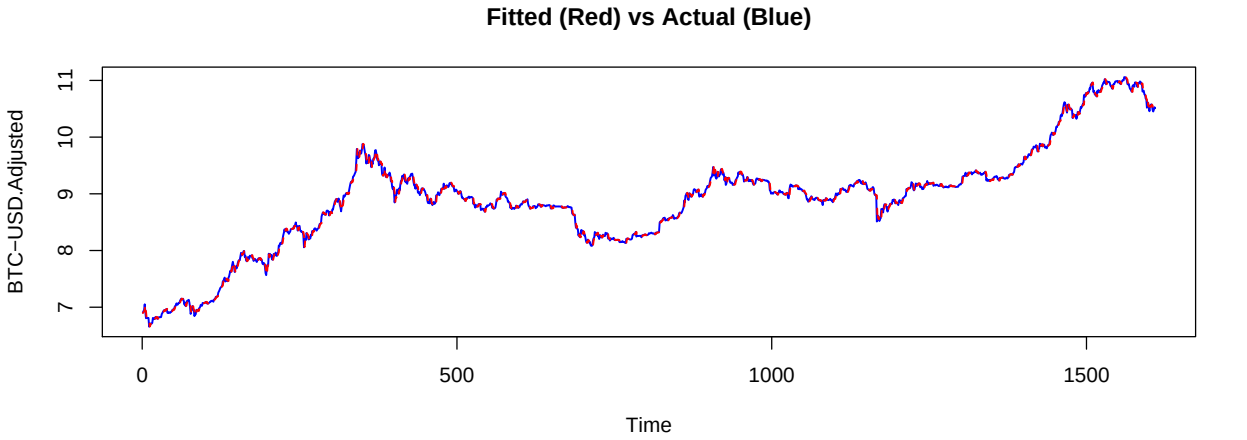


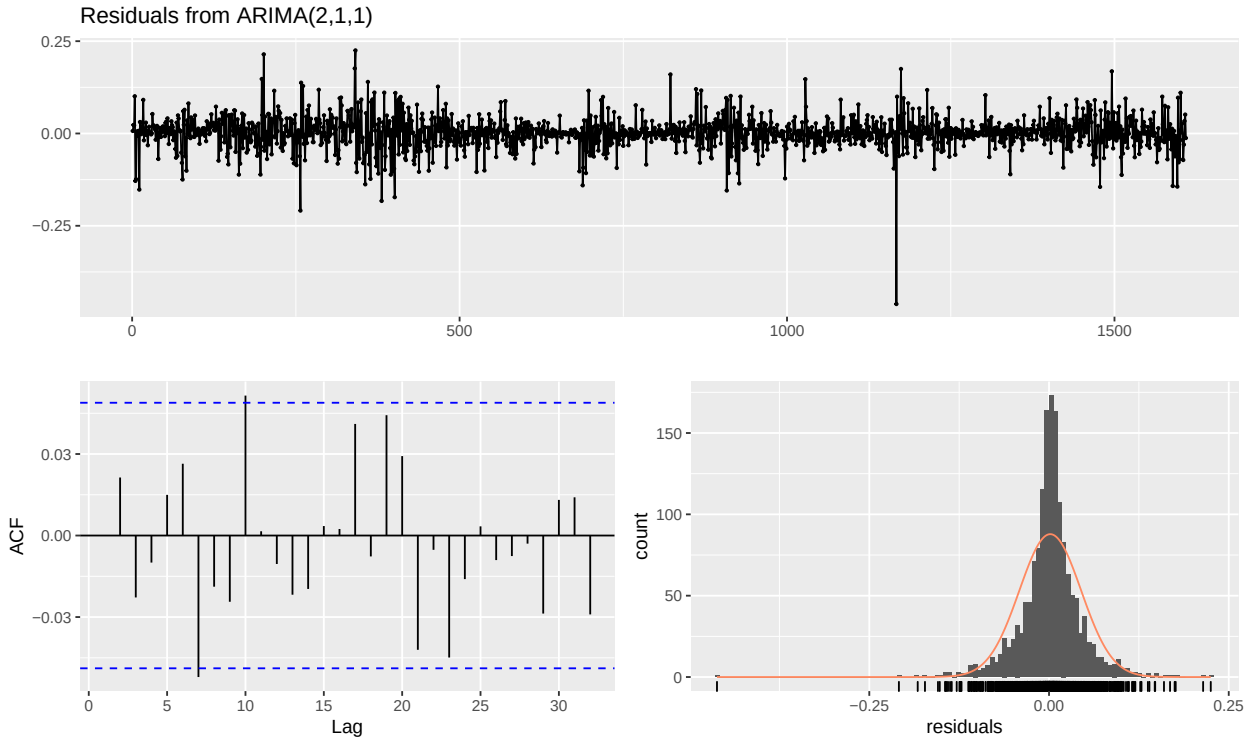
Upon reviewing the ACF plot for the log return, the cutoff for strong correlation is at 2 (or potentially 0) and the cutoff point of PACF indicates lag 1. The suggested model to fit ARMA model is ARMA(2,1). This will be further validated by comparing with grid-search method.

```
##                  AIC                BIC Specification
## 1 -5547.70978858816 -5520.79605633947         3,1,1
## 2 -5547.97629051928 -5515.67981182084         3,1,2
## 3 -5549.63141412985 -5528.10042833089         1,1,2
## 4 -5549.72656836021 -5528.19558256126         3,1,0
## 5 -5549.75625043726 -5538.99075753778         1,1,0
## 6 -5550.89738984375 -5534.74915049453         1,1,1
```

```
## 7 -5551.70728628242 -5535.55904693321          2,1,0
## 8 -5552.15840696638 -5530.62742116743          2,1,1
## 9 -5552.49785324462 -5525.58412099592          2,1,2
```

According to the grid search result, the best fit model specification is ARMA(3,1). However, compared to the initial model guess, ARMA(2,1), the difference in AIC & BIC is marginal. Hence, ARMA(2,1) will be considered as the pilot model for ARMA approach.






```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)
```

```
## Q* = 13.448, df = 7, p-value = 0.06192
##
## Model df: 3.   Total lags used: 10
```

Surprisingly, the actual vs fitted plot shows that the fitted values of model tend to follow the trend changes of the actual log price series. However,based on the Ljung-Box test and ACF plot of model residuals, it can be concluded that this ARMA model is not appropriate for forecasting since its residuals do not show white noise behavior although they are uncorrelated against each other.

Table 5: Variance Ratio Test

stat	sum
27.18773	2.301008

In addition, the variance ratio test result further supports the need of (G)ARCH type of modelling as the p-value is a lot greater than the significance level at 5%, meaning in-equal variance. Hence, the next section is going to delve into the (G)ARCH modelling techniques that can remedy this ARMA modelling.

2.2 (Generalised) Autoregressive conditional heteroskedasticity model (G)ARCH

The earlier ARIMA model showed a satisfactory result in the actual vs fitted tracking of log price series. However, the detailed analysis with first differenced series (log return) did not seem to capture the magnitude of volatility observed. The (G)ARCH model concerns time series with time-varying heteroskedasticity, where variance is conditional on the information existing at a given in time. This model has a foundation on making volatility clustering and will be effective in terms of capturing the empirical behaviour of volatility compared to ARIMA model. As a starting point, this report is going to explore the log return series with ARCH model and expand the model specification if applicable. In terms of model comparison, the performance metrics such as AIC and BIC are going to be examined as well as the pattern of conditional volatility compared to the movement of absolute log return series.

2.2.1 ARCH

ARCH model assumes that the conditional mean of the error term in a time series model is constant (zero) but the conditional variance is not. First, the specific test is required to understand if the time series has the ARCH effects. Hence, this report is taking ARCH LM-test that null hypothesis is no ARCH effects.

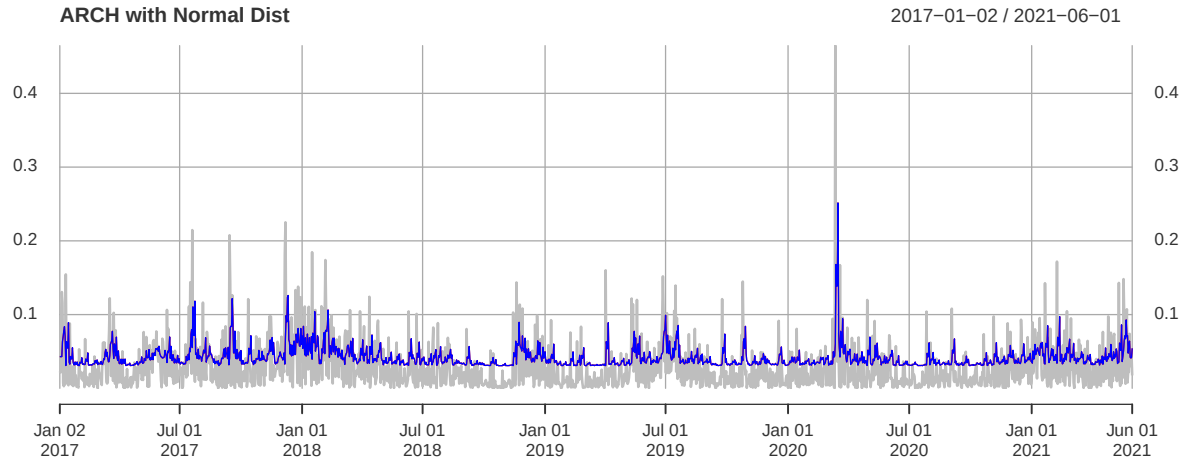
```
## Q(m) of squared series(LM test):
## Test statistic:  12.65589  p-value:  0.0003743839
## Rank-based Test:
## Test statistic:  55.48472  p-value:  9.414691e-14
```

The result of LM statistic, equal to 12.66 and the p-value is smaller than 5%. This indicates that the null hypothesis of no ARCH effects is rejected, concluding that the series has ARCH effects. This approves the ARCH modelling motivation. Using rugarch pcakge, the order of ARCH(p) model is going to be determined by iterating the model construction for $p = 1, 2, \ldots n$. (ARCH(1) is omitted due to error.)

Table 6: ARCH(p) Model Metrics

	AIC	BIC	SIC	HQIC
ARCH(2)	-3.486387	-3.472997	-3.486399	-3.481416
ARCH(3)	-3.497476	-3.480738	-3.497495	-3.491262
ARCH(4)	-3.564654	-3.544569	-3.564682	-3.557198

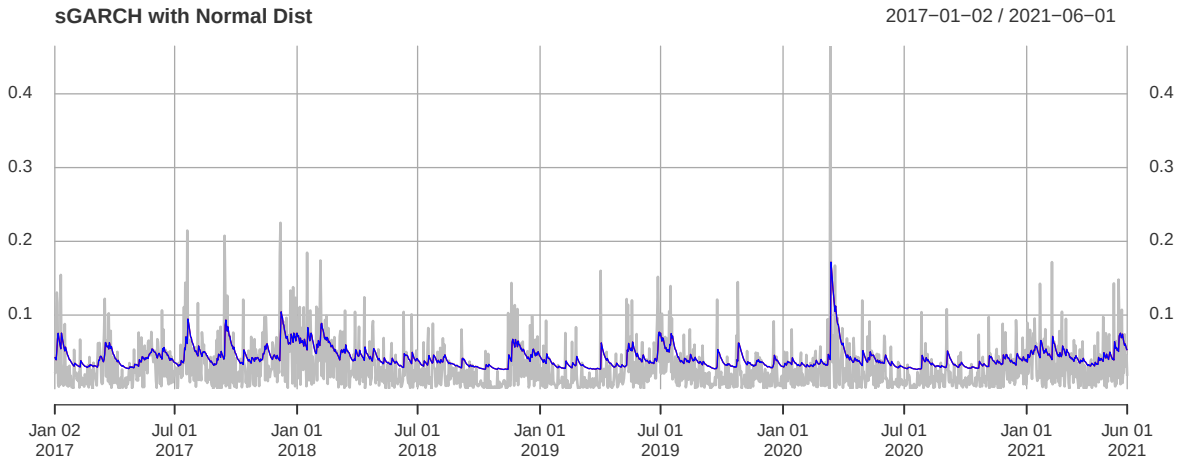
According to the iterative results of ARCH(1 ~ 4) models, the model evaluation metrics get better as the order of p increases. This suggests the higher order ARCH(p) or GARCH(1,1) model.



According to the conditional SD vs absolute log return series plot, it seems to capture the some of the largest historic volatility, but is not successful at replicating the most of significant historic volatility. The historic volatility seems to be more persistent. To overcome the problem of the persistence nature of volatility that a simple ARCH(p) model cannot capture, generalised ARCH (GARCH) model needs to be used. Hence, different versions of GARCH model specifications are going to be investigated in the next section.

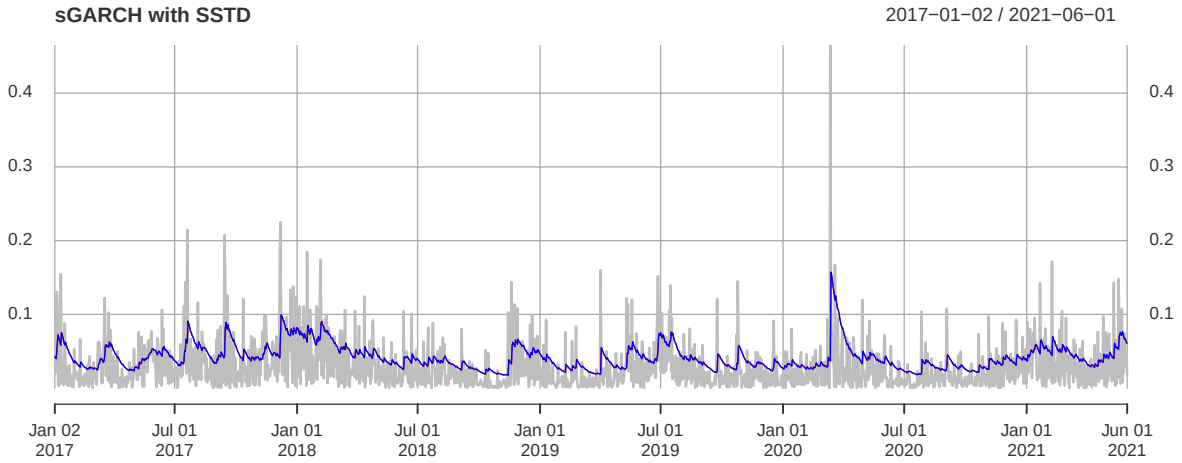
2.2.2 Standard GARCH

Again, using rugarch package, the Standard GARCH (1,1) model with normal distribution is constructed.



GARCH(1,1) still doesn't seem to track the historic volatility movement that well. Normal distribution is not consistent with stock market phenomenon. To account this, skewed-student t distribution (SSTD) is selected which tends to explain the stock price movement better.

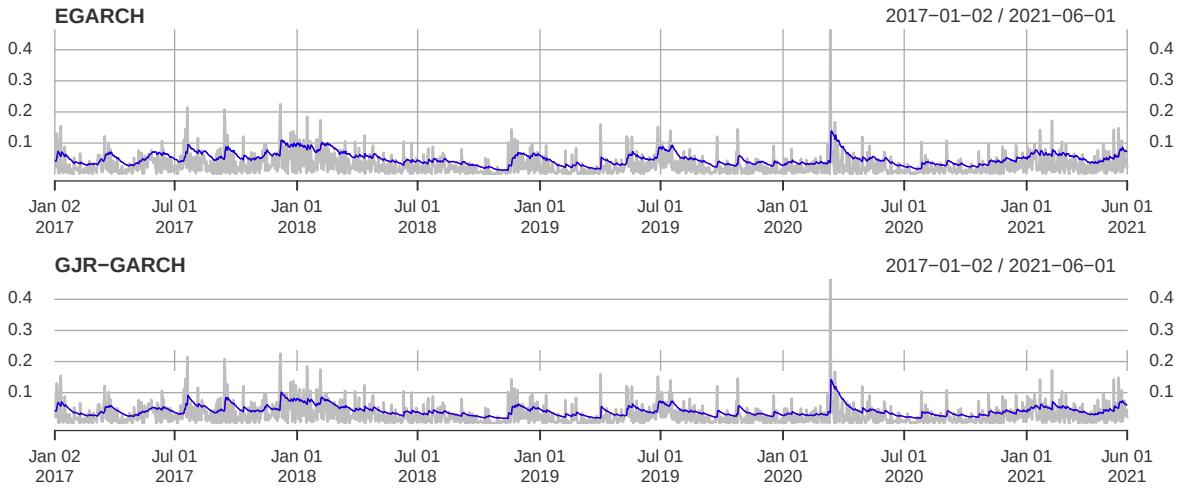
2.2.3 Standard GARCH with Skewed Student t distribution



Based on the information criteria results, GARCH(1,1) model with SSTD performs much better than the earlier models. This indicates that BTC price movement can be explained further with SSTD rather than the normal distribution.

In the market, the leverage effect is a very well-recognised phenomena that the near future volatility tends to be larger when the today's return is negative comparing to the positive return, meaning the negative returns induce the higher leverage and volatility. One way to remedy this is to use Exponential GARCH (EGARCH) model, which does not assume the symmetry property. Or the threshold GARCH, aka GJR-GARCH.

2.2.4 EGARCH / GJR-GARCH (Threshold GARCH)

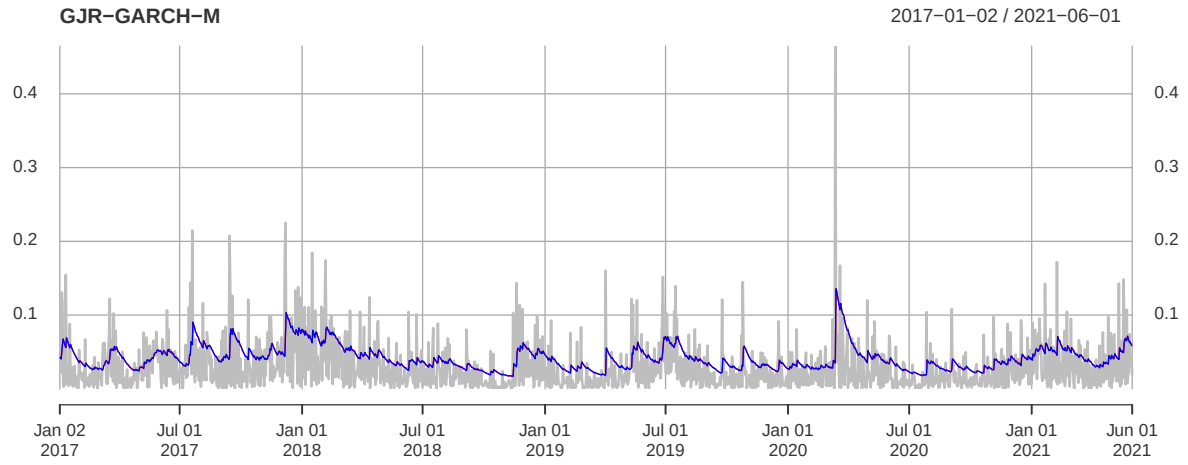


The conditional volatility curves for both models do not show a significant improvement in tracking the volatility clustering pattern better compared to the earlier models. To elaborate further, GARCH-in-mean model can be considered. In the finance theory, the return of stocks (BTC in this case), may depend on its volatility. To model this phenomena, the GARCH-in-mean (GARCH-M) model adds a heteroskedasticity term into the mean equation. The most widely-used and simplistic model is AR(1) model. The sign of AR parameter can be interpreted as following:

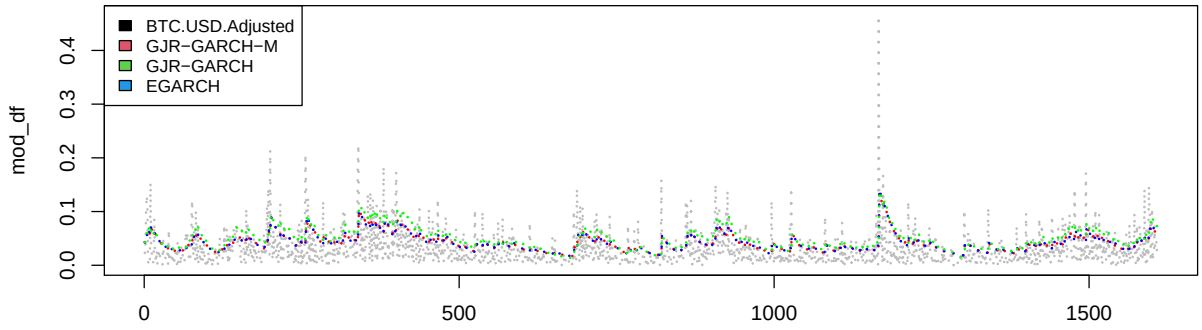
1. Positive: When the return is above its mean, the following sequence will get the momentum of this news and stay above mean as well.
2. Negative: When the return is below its mean, then a higher than average return is followed by a lower than average return. This means that the market shows an overreaction to the news.

Table 7: Model Performance Metrics

	AIC	BIC	SIC	HQIC
(Normal) GARCH(1,1)	-3.577977	-3.564587	-3.577989	-3.573006
(SSTD) GARCH(1,1)	-3.836887	-3.816802	-3.836914	-3.829430
EGARCH	-3.854058	-3.830626	-3.854096	-3.845359
GJR-GARCH	-3.836233	-3.812801	-3.836271	-3.827534
GJR-GARCH-M	-3.837085	-3.810305	-3.837134	-3.827143



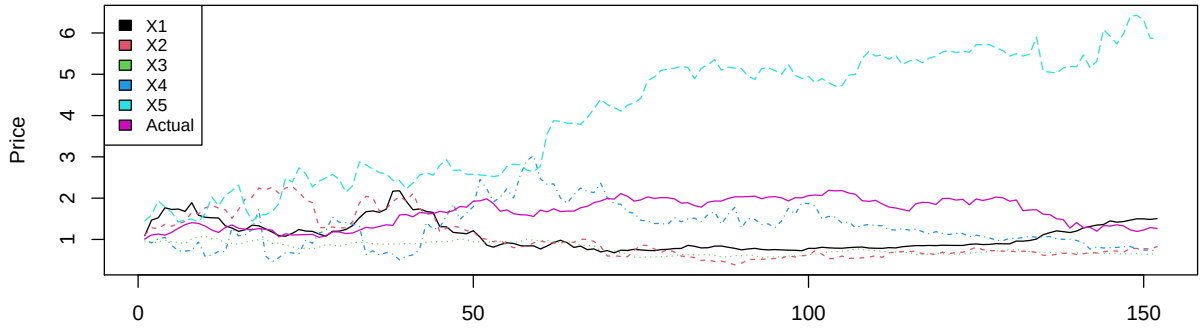
The GJR-GARCH-M model shows a little improvement compared to the earlier EGARCH and GJR-GARCH models. In order to achieve a robust modelling approach, top 3 models are going to be estimated and then compared with each other.



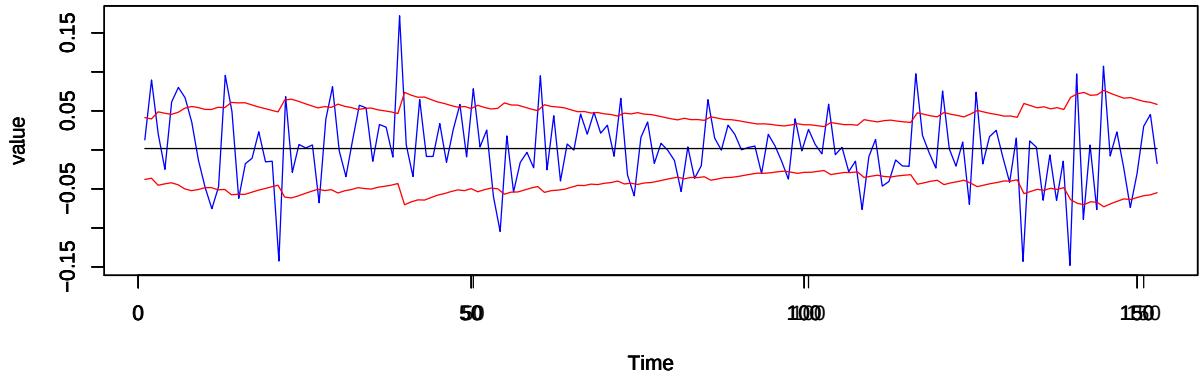
By inspecting the top 3 different models, it is notable to see that unlike the final model performance metrics table showed, the second-best GJR-GARCH model tends to be better at capturing the volatility in terms of the magnitude.

2.3 Forecasting the volatility

So far, the entire efforts have been concentrated on the selection of the appropriate model. In this section, the model forecasting & evaluation are going to be discussed. First of all, the data split between development and evaluation (train & test) is essential to ensure the un-biased modelling. In order to reflect the price volatility during COVID-19, the cohort date, 2020 Dec, is selected to include the one year horizon after the first recognition of COVID-19.



To evaluate the performance of the model in terms of the production (forecasting), 5 different simulated price series are generated. The forecast vs actual plot shows that the actual price series sits within the range of possible simulations that EGARCH process can generate. However, the caveat is that this doesn't necessarily mean the model has a great predictive power.



To understand whether the model can estimate the time series, n-step ahead forecast is used to compare with the actual time series of our interest. According to the chart, it seems like the model can cover the most of actual time series except those points with abnormal price movements.

2.4 Conclusion

In this report, different types of time series model have been explored to estimate the rapid & dramatic trend changes observed in BTC price. According to the result, GARCH type models showed the acceptable performance at capturing the observed fluctuations in volatility. However, the proposed GJR-GARCH model still does not seem to capture the entire dramatic price movements of BTC. This is one of the main challenges associated with the recent abnormal economic periods under COVID-19 as well as the unique nature of cryptocurrency market. In terms of future research, different modelling techniques including the Long-Short-Term-Memory (LSTM) model

and other advanced MLs such as XGBoost regression, SVM regression and others can be examined and compare the results to the conventional time series modelling approach.

3 Q3

In today's blog post, we are going to delve into the impact COVID-19 on the major global stock markets.

As seen in the news, almost every aspect of daily life and the global economy have been affected by this unprecedented global virus spread and economies across the world have been reflecting very different reactions according to each individual country's political measures and success of virus containment. On top of the virus dispersion, the unprecedented fiscal policies in major economies have changed the natural flow dynamics of financial market and the agents' behaviours that we used to know and understand. However, it is difficult to understand when and how these changes are happened.

The stock market index may not be the best indicator for understanding the global financial impact of COVID-19, but it is certainly the most representative and approachable measure of the financial market reaction compared to all the other finance-related indexes. Therefore, the stock market indexes are chosen to examine the following aspects:

1. Has COVID-19 increased (decrease) the market volatility?
2. Has COVID-19 caused the instability in the market movement prediction? In other words, has there been any structural breaks?
3. Can we still create the model to provide the consistent estimation of the price volatility after COVID?

Top 5 stock market indexes for today's blog are selected based on the global economic presence of the stock market within the continent. Asia: Nikkei225 / SSE Composite Index, Europe: DAX / FTSE100 and America: S&P500.

World-wide data for confirmed new cases and new deaths of COVID-19 comes from Our World in Data repository and COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. And all price time series of stock markets come from Yahoo Finance.

3.1 Data Summary Statistics

3.1.1 Confirmed Cases

Table 8: Summary Statistics: Confirmed Cases

	China	Germany	Japan	United.Kingdom	United.States	World
Mean	183.0	7460.00	1500.00	9100.00	6.73e+04	3.43e+05
Median	16.0	2980.00	710.00	3240.00	4.72e+04	2.92e+05
SD	907.0	9140.00	1810.00	13000.00	6.45e+04	2.51e+05

	China	Germany	Japan	United.Kingdom	United.States	World
Min	-1.0	0.00	0.00	-4790.00	0.00e+00	0.00e+00
Max	15100.0	49000.00	7910.00	68200.00	3.00e+05	9.06e+05
Skew	10.8	1.45	1.51	2.11	1.45e+00	3.19e-01
Kurt	159.0	4.53	4.52	7.39	4.32e+00	1.97e+00
#Obs	494.0	494.00	494.00	494.00	4.94e+02	4.94e+02

3.1.2 Confirmed Deaths

Table 9: Summary Statistics: Confirmed Deaths

	China	Germany	Japan	United.Kingdom	United.States	World
Mean	9.35	179.00	26.10	259.00	1200.00	7.15e+03
Median	0.00	60.50	11.00	76.00	959.00	6.27e+03
SD	63.20	263.00	33.40	358.00	984.00	4.30e+03
Min	0.00	-31.00	-1.00	0.00	0.00	0.00e+00
Max	1290.00	1730.00	248.00	1830.00	4480.00	1.79e+04
Skew	17.20	2.22	2.03	1.78	1.17	2.59e-01
Kurt	344.00	8.53	9.03	5.86	3.99	2.41e+00
#Obs	494.00	494.00	494.00	494.00	494.00	4.94e+02

Based on the summary statistics of confirmed cases & deaths of COVID-19, there are no outliers apart from the negative minimum values in certain countries. These negative minimum values in confired cases & deaths are corrections to rectify the known mistakes & errors. One caveat from this is that those countries (especially UK) with outliers may not result an accurate analysis. Hence, instead of utilising the series of individual country, the world aggregate series is going to be used as a proxy for COVID impact.

3.1.3 Stock Market Indexes (Pre COVID-19)

Table 10: Summary Statistics: Pre COVID-19

	GSPC.Adjusted	FTSE.Adjusted	DJI.Adjusted	N225.Adjusted	GDAXI.Adjusted	X000001.SS.Adjusted
Mean	0.00105	4.62e-04	7.83e-04	6.91e-04	8.72e-04	7.09e-04
Median	0.00135	8.25e-04	1.06e-03	9.03e-04	1.89e-03	4.68e-04
SD	0.00756	7.68e-03	7.58e-03	8.74e-03	8.92e-03	1.20e-02
Min	-0.03070	-3.71e-02	-3.14e-02	-3.10e-02	-3.83e-02	-5.47e-02
Max	0.02100	2.20e-02	2.02e-02	2.49e-02	2.78e-02	5.30e-02
Skew	-1.21000	-7.87e-01	-1.15e+00	-4.49e-02	-8.43e-01	-8.86e-02
Kurt	6.97000	6.00e+00	6.73e+00	3.88e+00	5.32e+00	7.03e+00
#Obs	213.00000	2.13e+02	2.13e+02	2.13e+02	2.13e+02	2.13e+02

3.1.4 Stock Market Indexes (Post COVID-19)

Table 11: Summary Statistics: Post COVID-19

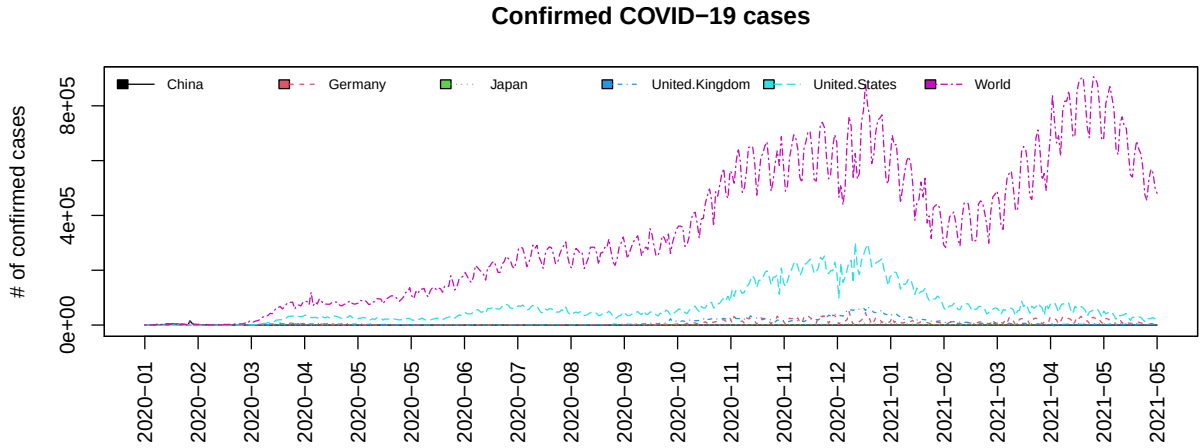
	GSPC.Adjusted	FTSE.Adjusted	DJI.Adjusted	N225.Adjusted	GDAXI.Adjusted	X000001.SS.Adjusted
Mean	5.82e-04	-3.91e-04	0.00033	5.61e-04	3.08e-04	5.70e-04
Median	1.81e-03	3.86e-04	0.00158	5.30e-04	7.07e-04	6.92e-04
SD	2.14e-02	1.81e-02	0.02270	1.66e-02	2.02e-02	1.32e-02
Min	-1.36e-01	-1.22e-01	-0.14800	-6.47e-02	-1.39e-01	-8.37e-02
Max	8.58e-02	8.30e-02	0.10200	7.44e-02	9.89e-02	7.27e-02
Skew	-1.35e+00	-1.20e+00	-1.39000	1.20e-01	-1.09e+00	-5.95e-01
Kurt	1.31e+01	1.25e+01	14.10000	6.39e+00	1.41e+01	1.15e+01
#Obs	2.90e+02	2.90e+02	290.00000	2.90e+02	2.90e+02	2.90e+02

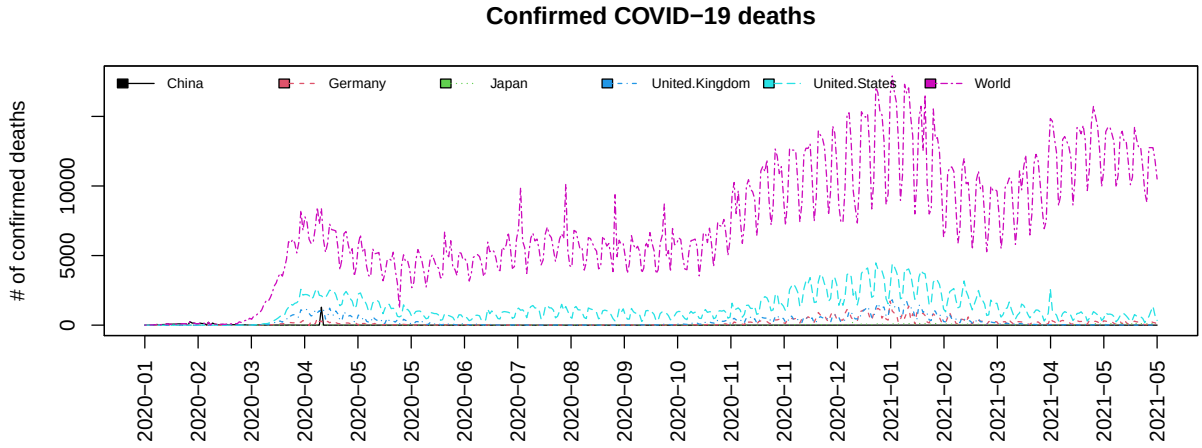
Table 12: Standard Deviation of Index

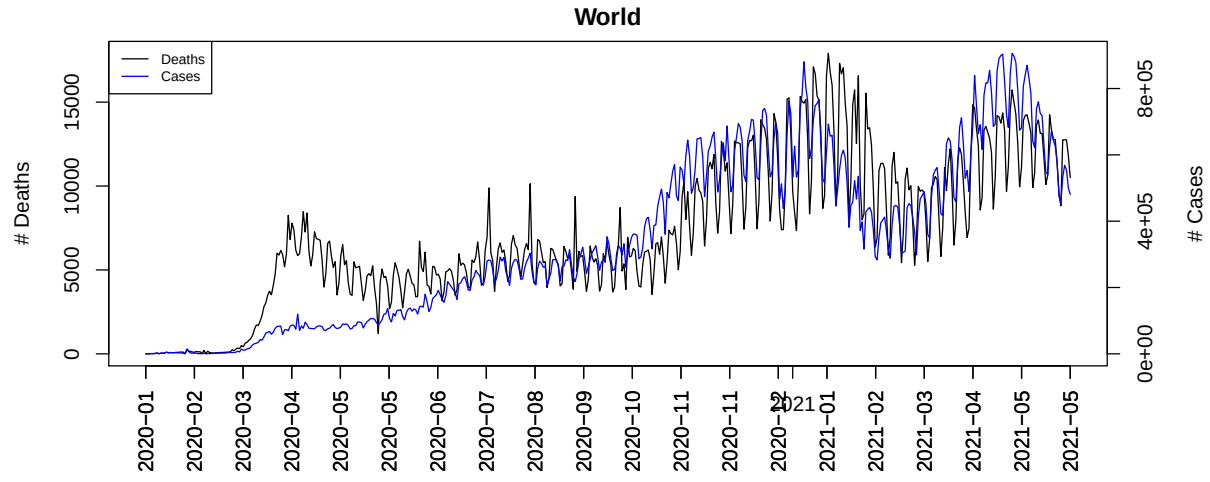
	(Post) SD	(Pre) SD	Post / Pre
GSPC.Adjusted	0.0214	0.00756	2.84
FTSE.Adjusted	0.0181	0.00768	2.35
DJI.Adjusted	0.0227	0.00758	3.00
N225.Adjusted	0.0167	0.00874	1.91
GDAXI.Adjusted	0.0202	0.00892	2.27
X000001.SS.Adjusted	0.0132	0.01200	1.10

Summary statistics of index returns are separated based on the cohort of COVID-19. Based on simple inspection, all the indexes show greater (e.g., mostly 2x larger, pre COVID GSPC: -3% vs post COVID GSPC: -13.6%) maximum draw-down as well as the lower mean return (e.g., pre COVID GSPC: 0.1% vs post COVID GSPC: 0.058%) in post COVID-19 compared to pre COVID-19. In addition, standard deviation of each index series increased dramatically as well. This indicates that there has been an adverse effect material enough to change the market characteristics across the globe and the market instability has increased after COVID-19.

3.2 Graphical Analysis

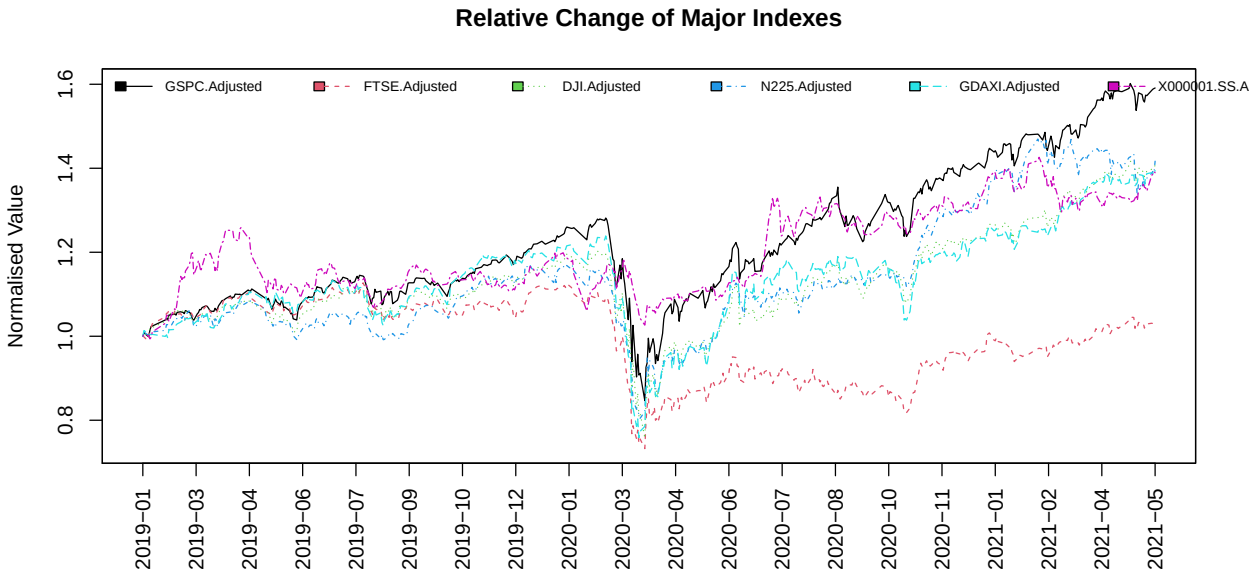






The plot of confirmed COVID-19 cases & deaths indicate that there is a lagging relationship between the confirmed cases and deaths. Intuitively, the confirmed cases are the leading factor of deaths and magnitude of deaths count is not as big as the confirmed cases. The first singificant

increase in both counts is at 2020-Mar then the confirmed cases gradually increase wherease the death toll remains the same level. The peak of confirmed cases is at 2020-Dec and 2020-Jan for the death toll. The number of cases & death both start to deflate right after the peak but re-bounce around 2021-Feb and 2021-Mar. The confirmed cases reach another peak during 2021-May whereas the death toll records the second largest peak at the same time of the year.



Based on the visual inspection on the major stock market movements, all the markets except SSE Composite Index crash in 2020-Mar. This coincides with the first increase of both confirmed cases and the death toll. The shock caused by COVID-19 caused severe falls in the stock markets around the world. According to the BBC news, on 2020-03-31, the Down Jones Industrial Average, S&P500 and FTSE 100 had plunged by 23%, 20% and 25% respectively, the biggest quarterly drops since 1987. In contrast to this, SSE Composite Index had a decline of 2.53% since the first case of COVID-19 was reported in the country. This could be the result of Chinese government's strong measures to contain the virus, which can be also seen in low number of confiremd cases & deaths of China in the earlier time series analysis. The notable point is that after the COVID shock in 2020-Mar, all the major indexes quickly recovered to pre-COVID level whereas FTSE100 still did not stay below the pre-COVID level until today.

The above visual inspection is very intuitive and simple for general understanding on the trends and changes. However, more robust and test-driven decisions and assessment would be a value adding point. Hence, in order to identify whether there has been a significant impact on the stock markets under COVID-19, aforementioned structural break tests are going to be examined.

3.3 Structural Break Test

Structural break test with regressions, also known as CHOW test is first introduced by Chow using F-statistics to identify the single break point with the assumption that the break-date is known. Since we are interested in estimating the timing of the breaks as well as the confidence intervals of them, we are going to use Bai and Perron's method for detecting a structural break instead of CHOW test. Bai and Perrorn proposed a method to estimate the breakpoints and the associated

confidence intervals in OLS regression with dynamic programming. Using this, it is possible to find the structural breakpoints that minimise the sum of squared residuals of the model with m+1 segments (optimal breakpoints). The model selection strategy is available in the package "strucchange".

There are different types of structural break tests that we can formulate based on differing assumptions:

1. level structural changes
2. trend structural changes

In general, financial time series like stock prices evolve over time. Hence the trend or polinomial fit structural changes would be more relevant than the no-trend fit. However, regardless of practicality of the assumption, we start with the simplest form, level structural changes for 5 major stock indexes which follows the equation below:

$$\log(Y_{Index}) \sim Constant$$

As shown in the formula, this type of assumption would suit better with the return series, which is believed to be the 0 mean series. Code for the level structural break test is shown below:

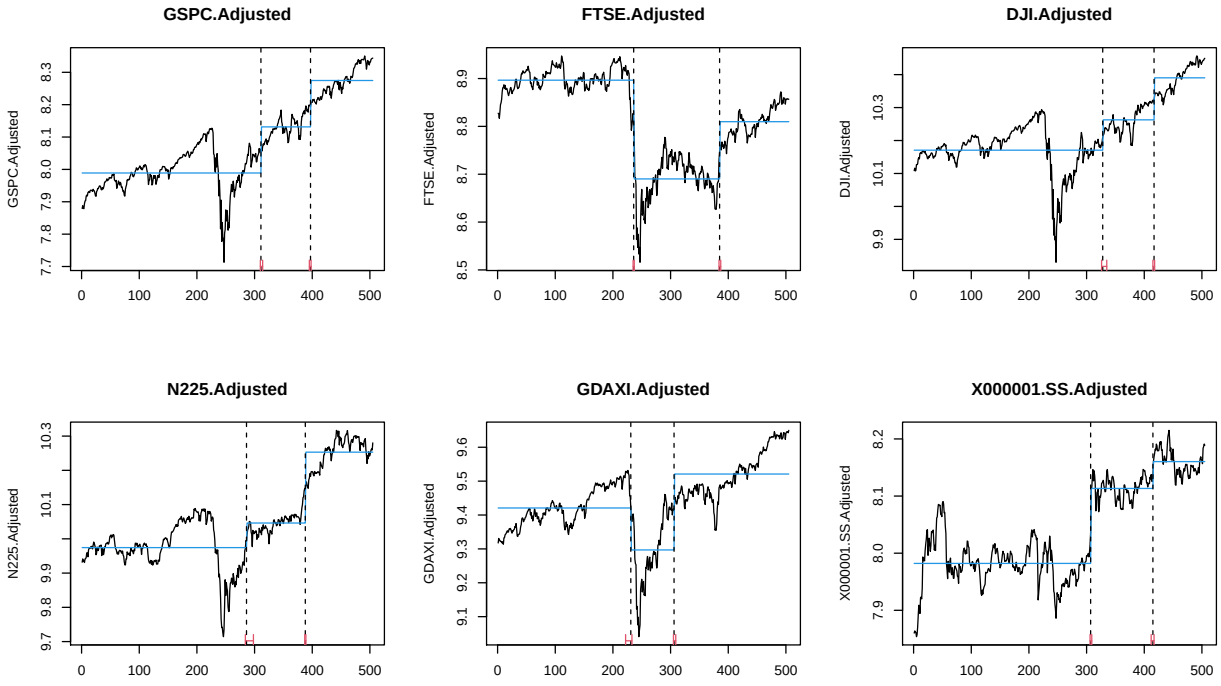


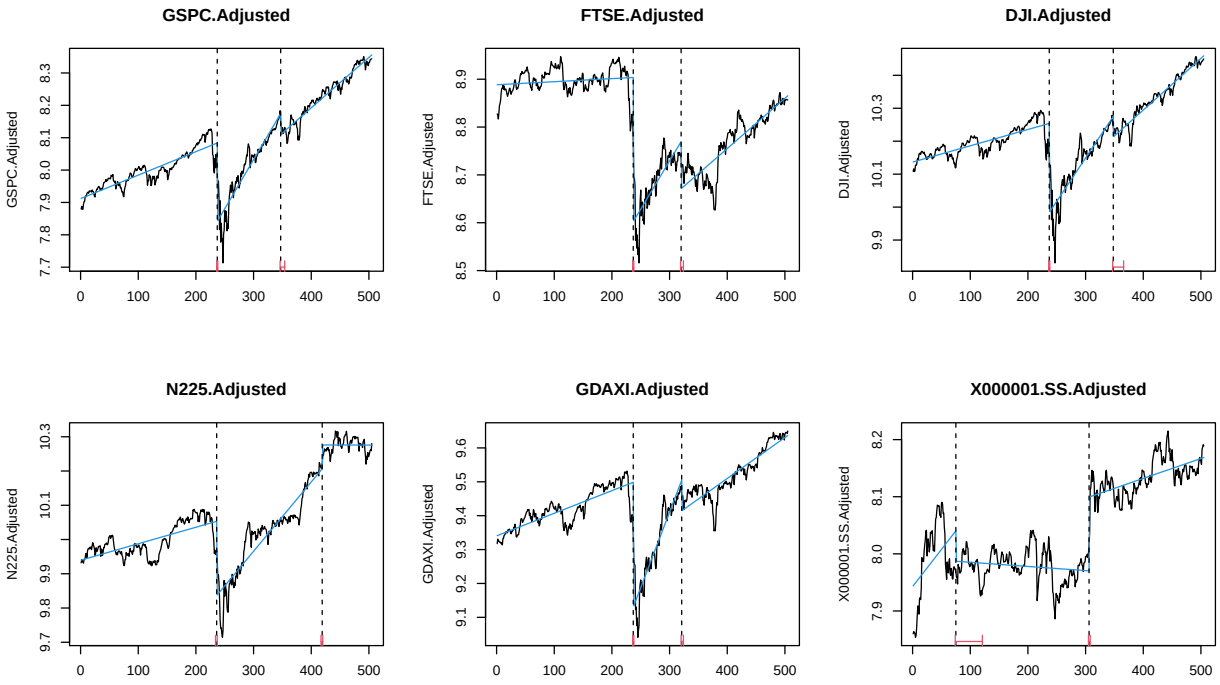
Table 13: Level Structural breakpoints & dates

GSPC.Adjusted	FTSE.Adjusted	DJI.Adjusted	N225.Adjusted	GDAXI.Adjusted	X000001.SS.Adjusted
2020-07-09	2020-03-05	2020-08-05	2020-05-29	2020-02-27	2020-07-02
2020-11-30	2020-11-10	2021-01-05	2020-11-13	2020-07-01	2020-12-30

The above results show the plots of the observed and fitted time series, along with confidence intervals for the breakpoints. We examined 2 major breakpoint dates and according to the identified breakpoints & dates, COVID-19 shock in 2020-Mar is confirmed to be the structural breakpoint for only two indexes (FTSE & DAX). This could be due to the aforementioned no-trend assumption of our test. Referring back to the visual inspection of indexes, there is a clear upward linear (or perhaps non-linear) trend. To remedy this, we introduce the linear trend structural change method, which is believed to be a more realistic assumption.

The equation and the code for the trend structural break test follow:

$$Y_{Index} \sim Trend$$



Similar to the level structural change case, most of the indexes are reaching the minimum BIC when breakpoints (m) = 2. Hence, we explore the 2 major breakpoint dates.

Table 14: Trend Structural breakpoints & dates

GSPC.Adjusted	FTSE.Adjusted	DJI.Adjusted	N225.Adjusted	GDAXI.Adjusted	X000001.SS.Adjusted
2020-03-06	2020-03-06	2020-03-06	2020-03-05	2020-03-06	2019-06-03
2020-09-03	2020-07-22	2020-09-04	2021-01-07	2020-07-27	2020-07-01

According to the identified breakpoints & dates, COVID-19 shock in 2020-Mar is confirmed to be the significant structural breakpoint for all the indexes except SSE Composite Index. As mentioned in the earlier section, China was the first country to achieve economic recovery amid COVID-19 events and removed the major city lock-downs, backed by its rapid and active counter actions

against the spread of virus. Hence, this unique background of China could have mitigated the COVID-19 shock in 2020-Mar.

These finding sheds a light on the following aspects:

1. Due to the severe COVID-19 shock, the presence of structural break is confirmed.
2. COVID-19 shock may be a systematic event across the globe but it is not necessarily an inevitable risk considering the case of China.
3. Due to the structural break, existing statistical models & analysis that have been used pre COVID-19 will be obsolete, proposing a potential model risk for major businesseses.
4. The financial market that we used to know, we think we understand, will be no longer the same. This proposes a huge threat at the same time a great opportunity.

In the Appendix.A, the polinomial fit structural change analysis is carried out and further advocates the hypothesis that COVID-19 shock has introduced the significant structural break across the countries. In the next section, we attempt to model the price volatility using GARCH.

4 Appendix: All code for this report (Except Q1)

```
# Library Setup
library(readxl)
library(data.table)
library(Hmisc)
library(FSA)
library(strucchange)
library(tseries)
library(forecast)
library(quantmod)
library(knitr)
library(MTS)
library(urca)
library(tidyverse)
library(vrtest)
library(moments)
library(rugarch)
simulate_garch = function(params, n, d)  {
  #' GARCH(1,1) simulation function for given distribution.
  #' @param params vector. The sequence of parameters, a0, a1, b1 respectively
  #' @param n int. # of simulation size
  #' @param d vector. Sample population from the designated distribution
  a0 = params[1]
  a1 = params[2]
  b1 = params[3]
  sigma = a0 / (1 - a1 - b1)
  x0 = sqrt(sigma) * d[1]
```

```
  x = c(x0)
  for (i in 2:n) {
    sigma_new = a0 + a1 * x0^2 + b1 * sigma
    x1 = sqrt(sigma_new) * d[i]
    x = c(x, x1)
    x0 = x1
    sigma = sigma_new
  }
  return(x)
}

# Simulate GARCH (1,1) Series with normal distribution
x = simulate_garch(c(0.1, 0.6, 0.3) , 100, rnorm(100))
set.seed(1)
gmle = function(x, params) {
  #' GMLE function in the paper.
  #' @param x vector. An array to be estimated
  #' @param params vector. The sequence of parameters, a0, a1, b1 respectively
  a0 = params[1]
  a1 = params[2]
  b1 = params[3]
  n = length(x)
  t_sig = c(a0/(1-b1) + a1 * x[1]^2) # Equation (2.5)
  t_sig = c(t_sig, a0 / (1 - b1) + a1 * x[2:(n-1)]^2 + a1 * cumsum(x^2*b1^c(n:1))[1:(n-2)])
  ll = sum(x[2:n]^2/t_sig + log(t_sig)) # Equation (2.4) in the paper
  if (a0>=0 && a1 >= 0 && b1 >= 0 && (a1+b1) < 1) { # Required constraint for GARCH equation
    return(ll)
  } else {return(1e+09)}}

garch_gmle <- function(x) {
  #' Optimisation for GMLE.
  #' @param x vector. An array to be estimated
  params = c(0.1, 0.1, 0.1) # initial parameters for optimisation
  result = optim(params, gmle, x=x, method = c("BFGS")) # BFGS Algorithm for minimisation
  res = data.frame(result$par)
  row.names(res) = c("alpha0","alpha1","beta")
  res["-LL", ] = result$value
  return(res)}
kable(garch_gmle(x), caption="GARCH MLE")
lade_mle = function(x, params) {
  #' LADE function in the paper.
  #' @param x vector. An array to be estimated
  #' @param params vector. The sequence of parameters, a0, a1, b1, c0 respectively
  a0 = params[1] # c in the paper
  a1 = params[2] # b1 in the paper
  b1 = params[3] # a1 in the paper
  c0 = params[4] # C0 in the paper
```

```
  gamma = a0 / c0
  beta = a1 / c0
  n = length(x)
  s2 = c(gamma/(1-b1) + beta * x[1]^2) # Equation (2.6)
  s2 = c(s2, gamma / (1 - b1) + beta * x[2:(n-1)]^2 + beta * cumsum(x^2*b1^c(n:1))[1:(n-2)])
  ll = sum(abs(log(x[2:n]^2) - log(s2)))
  if (a0>=0 && a1 >= 0 && b1 >= 0 && a1+b1 < 1 && c0 > 0) { # Constraint for GARCH
    return(ll)
  } else {
    return(1e+09)}}

garch_lade <- function(x) {
  #' Optimisation for LADE.
  #' @param x vector. An array to be estimated
  params = c(0.1, 0.1, 0.1, 0.1) # initial parameters for optimisation
  result = optim(params, lade_mle, x=x, gr=NULL, method = c("L-BFGS-B"))
  res = data.frame(result$par)
  row.names(res) = c("a0","a1","b1", "C0")
  res["-LL", ] = result$value
  return(res)}

laplce_rv <- function(n, mu=0, b=1) {
  #' Laplace r.v. generator, source: Wikipedia
  #' @param n int. A number of samples
  #' @param mu float. Location parameter
  #' @param lam float. Scale parameter
  p = runif(n)
  x1 = b * log(2 * p) + mu
  x2 = -b * log(2 * (1-p)) + mu
  x1[x1>mu] = 0
  x2[x2<=mu] = 0
  return(x1 + x2)}
# Simulate GARCH (1,1) Series with normal distribution
lp_x = simulate_garch(c(0.6, 0.7, 0.2), 100, laplce_rv(100))
kable(garch_lade(lp_x), caption="GARCH LADE")
skewed_t_rv <- function(n, dof) {
  #' Skewed t r.v. generator
  #' @param n int. A number of samples
  #' @param dof int. Degrees of freedom
  v0 = abs(rnorm(n))
  v1 = rnorm(n)
  v2 = rchisq(n, dof)
  (0.8 * v0 + 0.6 * v1) / sqrt(v2 / dof)
}
n = 1000
cases = list(rnorm(n), rt(n, 3), skewed_t_rv(n, 6), skewed_t_rv(n, 3), laplce_rv(n, mu=0, b=1),
names(cases) = c("N(0,1)", "t with DoF=3", "Skewed t with DoF=6", "Skewed t with DoF=3", "Laplac
```

```
ee_func = function(param_hat, param_r) {
  #' Estimation error function.
  #' @param param_hat vector. A vector of estimated parameters
  #' @param param_r vector. A vector of true parameters
  return(abs(param_hat[2]/param_hat[1]-param_r[2]/param_r[1]) + abs(param_hat[3]-param_r[3]))}
params = c(0.5, 0.6, 0.3)
res = matrix(nrow=length(cases), ncol=2)
for (case_n in 1:length(cases)){
  d = cases[[case_n]] # Each case distribution
  e_gmle = c()
  e_lade = c()
  for (i in 1:5) {
    x = simulate_garch(params, n, d)
    gmle_res = garch_gmle(x)
    lade_res = garch_lade(x)
    eps_gmle = ee_func(as.vector(gmle_res[,1]), params)
    eps_lade = ee_func(as.vector(lade_res[,1]), params)
    e_gmle = c(e_gmle, eps_gmle)
    e_lade = c(e_lade, eps_lade)
  }
  res[case_n, ] = c(mean(e_gmle), mean(e_lade))
}
res = data.frame(res)
row.names(res) = names(cases)
colnames(res) = c("Epsilon_GMLE", "Epsilon_LADE")
kable(res, caption="Estimation Error by Distribution")
btc = na.omit(getSymbols("BTC-USD", src = 'yahoo', from = "2014-09-17", to = "2021-06-02", auto.
price = window(btc, start = "2017-01-01")[,6]
logprice = log(price)
btc.ts.log = log(ts(price, start=c(2014,09,17), frequency=252))
par(mfrow=c(1,2))
plot(price, main="Price")
plot(logprice, main="Log Price")
summary_stat = function(df){
  metrics = c("Mean","Median","SD", "Min", "Max","Skew", "Kurt", "#Obs")
  met_func = c(mean, median, sd, min, max, skewness, kurtosis, length)
  tab = matrix(nrow= length(metrics),
               ncol=dim(df)[2],
               dimnames = list(metrics,colnames(df)))
  for (i in 1:length(metrics)){
    tab[i,] = apply(df, 2, met_func[[i]])}
  return(signif(tab, 3))}
kable(summary_stat(price), caption="Summary Statistics: BTC Price")
decomp_analysis = function(ts_v) {
  #' Decomposition plot (Assess trend, seasonality and remainder)
  ts_v %>% decompose(type="additive") %>%
  autoplot() + xlab("Year") +
```

```
  ggtitle("Classical additive decomposition of log return")
}
stationarity_analysis = function(ts_v){
  #' Stationarity Test: ADF Test / KPSS
  print(summary(ur.kpss(ts_v)))
  return(adf.test(ts_v, k=12))
}
decomp_analysis(btc.ts.log) # Decomposition analysis for model identification
stationarity_analysis(btc.ts.log)
logret = diff(logprice) # Log Return
logret = logret[!is.na(logret)] # Omit NA Values
logret.ts = ts(logret, frequency = 252) # TS Object
decomp_analysis(logret.ts) # Decomposition Analysis
stationarity_analysis(logret.ts) # Stationarity test
Box.test(logret, type="Ljung-Box")
par(mfrow=c(1,2))
acf(logret)
pacf(logret)
grid_search_sarima = function(ts_v, pqd_terms) {
  #' Exhaustive approach of derive multiplicative seasonal ARIMA model.
  #' Iterate through each P, Q terms of non-seasonal components. (D is fixed.)
  #' @param ts_v (vector): A Time Series vector.
  #' @param pqd_terms (vector): Initial P, Q, D elements for non-seasonal component.
  c_pqd = expand.grid(seq(max(pqd_terms[1]-1, 0), pqd_terms[1] + 1), pqd_terms[2],
                              seq(max(pqd_terms[3]-1, 0), pqd_terms[3] + 1))
  mod_fit = data.frame(AIC = numeric(0),BIC = numeric(0), Specification = numeric(0))
  count = 1
  for (i in 1:dim(c_pqd)[1]){
    skip_to_next <- FALSE
    # Error catcher for ARIMA fitting (In case of optimisation error)
    mod = tryCatch(Arima(ts_v, order = as.numeric(c_pqd[i,])),
                   error = function(e) { skip_to_next <<- TRUE})
    if(skip_to_next) {next}
    mod_fit[count, ] = c(mod$aic, mod$bic, paste(as.character(c_pqd[i,]), collapse=","))
    count = count + 1}
  mod_fit = mod_fit %>% arrange(AIC)
  return(mod_fit)}
grid_search_sarima(logprice, c(2,1,1))
best_arma = Arima(logprice, order=c(2,1,1))
best_arma.fitted = xts(fitted(best_arma), index(logprice))
plot(ts(logprice), type="l", lwd=1.5, col="blue", main="Fitted (Red) vs Actual (Blue)")
lines(ts(best_arma.fitted), lwd=1.5, lty=2, col="red")
checkresiduals(best_arma) # Residual analysis
kable(data.frame(Auto.VR(logret)), caption = "Variance Ratio Test")
archTest(ts(logret), 1)
garch_modeler = function(model, garchorder, armaorder, dist){
  #' @param garchorder: GARCH(p,q) order
```

```
  #' @param armaorder: GARCH-in-Mean model order
  #' @param dist: Innovation distribution
  spec = ugarchspec(mean.model = list(armaOrder = armaorder),
                    variance.model = list(model=model, garchOrder = garchorder),
                    distribution.model = dist)
  return(ugarchfit(data=logret, spec=spec))}
arch_orders = c("ARCH(2)", "ARCH(3)", "ARCH(4)")
arch.res = matrix(nrow = length(arch_orders), ncol=4,
                  dimnames=list(arch_orders,  c("AIC","BIC","SIC","HQIC")))
for (i in 1:length(arch_orders)){
  model.arch = garch_modeler("sGARCH", c((i+1),0), c(0,0), dist="norm")
  arch.res[i,] = as.vector(infocriteria(model.arch))}
arch.res = data.frame(arch.res)
kable(arch.res, caption="ARCH(p) Model Metrics")
plot_conditional_vol = function(model, main){
  plotvol = plot(abs(logret), col = "gray", main=main)
  plotvol = addSeries(sigma(model), col = "red", on = 1)
  plotvol = addSeries(sigma(model), col = "blue", on = 1)
  return(plotvol) }
plot_conditional_vol(model.arch, "ARCH with Normal Dist")
sgarch =  garch_modeler("sGARCH", c(1,1), c(0,0), "norm") # Standard GARCH
metrics = data.frame(matrix(ncol=4,nrow=0, dimnames=list(c(), c("AIC","BIC","SIC","HQIC"))))
metrics["(Normal) GARCH(1,1)", ] =  infocriteria(sgarch)
plot_conditional_vol(sgarch, "sGARCH with Normal Dist")
stdgarch =  garch_modeler("sGARCH", c(1,1), c(0,0), "sstd") # T-distribution GARCH
metrics["(SSTD) GARCH(1,1)", ] =  infocriteria(stdgarch)
plot_conditional_vol(stdgarch, "sGARCH with SSTD")
egarch =  garch_modeler("eGARCH", c(1,1), c(0,0), "sstd") # EGARCH
metrics["EGARCH", ] =  infocriteria(egarch)
gjrgarch =  garch_modeler("gjrGARCH", c(1,1), c(0,0), "sstd") # GJR-GARCH
metrics["GJR-GARCH", ] =  infocriteria(gjrgarch)
par(mfrow=c(2,1))
plot_conditional_vol(egarch, "EGARCH")
plot_conditional_vol(gjrgarch, "GJR-GARCH")
gjrgarchm =  garch_modeler("gjrGARCH", c(1,1), c(1,0), "sstd") # GJR-GARCH IN MEAN
metrics["GJR-GARCH-M", ] =  infocriteria(gjrgarchm)
kable(metrics, caption="Model Performance Metrics")
plot_conditional_vol(gjrgarchm, "GJR-GARCH-M")
model_list = c(gjrgarchm, gjrgarch, egarch)
names(model_list) = c("GJR-GARCH-M", "GJR-GARCH", "EGARCH")
c = 1
mod_df = data.frame("Abs Ret" = abs(logret))
for (i in c(1,2,3)) {
    sig = sigma(model_list[[i]])
    colnames(sig) = names(model_list)[[i]]
    if (c==1) {
      msigma = sig} else {
```

```
        msigma = merge(msigma, sig)}
  c = c + 1
  mod_df[, names(model_list)[[i]]] = sig}
matplot(mod_df, type="l", col=c("grey", "red","blue", "green"), lwd=2, lty=3)
legend("topleft", colnames(mod_df),col=seq_len(ncol(mod_df)),cex=0.8,fill=seq_len(ncol(mod_df)))
train = logret["/2020-12"] # Train & Test split
test = logret["2021-1/"]
best_m = ugarchspec(mean.model = list(armaOrder = c(0,0)), # BEST MODEL
                        variance.model = list(model = "gjrGARCH"),
                        distribution.model = "sstd")
garchfit = ugarchfit(data = train, spec = best_m)
simgarchspec = best_m
setfixed(simgarchspec) = as.list(coef(garchfit)) # Simulation setting
simgarch = ugarchpath(spec = simgarchspec, m.sim = 5, n.sim = length(test), rseed =349)
simret = data.frame(fitted(simgarch)) # Simulated returns
simret$Actual = test # Actual returns
cumul_prices = exp(apply(simret, 2, "cumsum")) # Cumulative return of normal price
matplot(cumul_prices, type="l", ylab="Price") # Visualisation
legend("topleft", colnames(cumul_prices),col=seq_len(ncol(cumul_prices)),cex=0.8,fill=seq_len(nc
model.garch.fit = ugarchfit(data=c(train,test), spec=best_m, out.sample = length(test),  solver
modelfor = ugarchforecast(model.garch.fit, data=NULL, n.ahead=1, n.roll=length(test), out.sample
results1 = modelfor@forecast$seriesFor[1,] + modelfor@forecast$sigmaFor[1,]
results2 = modelfor@forecast$seriesFor[1,] - modelfor@forecast$sigmaFor[1,]
ylim <- c(min(test), max(test))
plot.ts(test , col="blue", ylim=ylim, ylab="value")
par(new=TRUE)
modelfor@forecast$seriesFor[1,]  %>% plot.ts(ylim=ylim)
par(new=TRUE)
plot.ts(results1, col="red", ylim=ylim, ylab="value")
par(new=TRUE)
plot.ts(results2, col="red", ylim=ylim, ylab="value")
read_csv = function(path){
  df = read.csv(path)
  df[is.na(df)] = 0
  return(xts(df[, -1], order.by = as.Date(df[,1])))}
cov_case_xts = read_csv("/Users/joonkang/Library/Mobile Documents/com~apple~CloudDocs/Group/Stud
cov_death_xts = read_csv("/Users/joonkang/Library/Mobile Documents/com~apple~CloudDocs/Group/Stu
stock_list = c("^GSPC", "^FTSE", "^DJI", "^N225", "^GDAXI", "000001.SS")
for (i in 1:length(stock_list)){
    dat = getSymbols(stock_list[i], src = 'yahoo', from = "2019-01-24", to = "2021-06-01", auto.
    dat = na.omit(dat)[, 6]
    if (i==1){
      prices = dat
    } else{
      prices = merge(prices, dat, by=0, all=TRUE)}
    prices = prices[ ,!(names(prices) %like% "by")]
    prices = na.omit(prices)}
```

```
prices.pre = prices["/2020-01-22"]
prices.post = prices["2020-01-23/"]
kable(summary_stat(cov_case_xts), caption="Summary Statistics: Confirmed Cases")
kable(summary_stat(cov_death_xts), caption="Summary Statistics: Confirmed Deaths")
ret.pre = diff(prices.pre)/prices.pre[-1]
kable(summary_stat(ret.pre), caption="Summary Statistics: Pre COVID-19")
sd.pre = matrix(ncol=1, nrow=6)
for (i in 1:dim(ret.pre)[2]){
  stdv = round(sd(ret.pre[, i]), 5)
    sd.pre[i,] = stdv}
row.names(sd.pre) = colnames(ret.pre)
colnames(sd.pre) = "(Pre) SD"
ret.post = diff(prices.post)/prices.post[-1]
kable(summary_stat(ret.post), caption="Summary Statistics: Post COVID-19")
sd.post = matrix(ncol=1, nrow=6)
for (i in 1:dim(ret.post)[2]){
  stdv = round(sd(ret.post[, i]), 5)
  sd.post[i,] = stdv}
row.names(sd.post) = colnames(ret.post)
colnames(sd.post) = "(Post) SD"
diff = sd.post/sd.pre
colnames(diff) = "Post / Pre"
sd.summary = signif(cbind(sd.post, sd.pre, diff), 3)
kable(sd.summary, caption="Standard Deviation of Index")
plot_df = function(df, ylab, main){
  d = as.Date(index(df))
  plot(as.zoo(df), ylab=ylab, main=main, col=seq_len(ncol(df)), screens = 1, lty = 1:dim(df)[2],
  axis.Date(1, at = seq(d[1], d[length(d)], length.out=20),
            format= "%Y-%m", las = 2)
  legend("topleft", col=seq_len(ncol(df)), colnames(df),
         lty = 1:length(df), cex = 0.7, fill=seq_len(ncol(df)),
         xpd = TRUE, horiz = TRUE, bty = "n")}
par(mfrow=c(2,1))
plot_df(cov_case_xts, ylab="# of confirmed cases", main="Confirmed COVID-19 cases")
plot_df(cov_death_xts, ylab="# of confirmed deaths", main="Confirmed COVID-19 deaths")
par(mar = c(5,5,2,5)) # Leaves some space for the second axis
d = as.Date(index(cov_death_xts))
plot(index(cov_death_xts), as.data.frame(cov_death_xts)$World, type="l", ylab="# Deaths", main="
axis.Date(1, at = seq(d[1], d[length(d)], length.out=20),format= "%Y-%m", las = 2)
par(new=T)
plot(index(cov_case_xts), as.data.frame(cov_case_xts)$World, type="l",axes=F, xlab=NA, ylab=NA,c
axis.Date(1, at = seq(d[1], d[length(d)], length.out=20),format= "%Y-%m", las = 2)
axis(side = 4)
mtext("# Cases", side = 4, line = 3)
legend("topleft", col=c("black", "blue"), c("Deaths", "Cases"), lty = c(1,1), cex = 0.7)
prices.norm = data.frame(prices) %>%mutate_if(is.numeric, list(~ ./ first(.)))
plot_df(as.zoo(prices.norm), ylab="Normalised Value", main="Relative Change of Major Indexes")
```

```
break_points = data.frame(matrix(ncol=dim(prices)[2], nrow=2,
                                 dimnames=list(c(), colnames(prices))))
par(mfrow=c(2,3))
for (i in colnames(prices)){
tests = breakpoints((ts(log(prices[,i])) ~ 1), breaks=2)
b_dates = breakdates(tests, breaks = 2)
break_points[,i]= index(prices)[b_dates]
plot(ts(log(prices[,i])), main=i, xlab="")
lines(fitted(tests, breaks = length(tests$breakpoints)), col = 4, xaxt="n")
lines(confint(tests, breaks = length(tests$breakpoints)), xaxt="n")
}
kable(break_points, caption = "Level Structural breakpoints & dates")
l = dim(prices)[1]
tt = 1:l

break_points = data.frame(matrix(ncol=dim(prices)[2], nrow=2,
                                 dimnames=list(c(), colnames(prices))))
par(mfrow=c(2,3))
for (i in colnames(prices)){
tests = breakpoints((ts(log(prices[,i])) ~ tt), breaks=2)
b_dates = breakdates(tests, breaks = 2)
break_points[,i]= index(prices)[b_dates]
plot(ts(log(prices[,i])), main=i, xlab="")
lines(fitted(tests, breaks = length(tests$breakpoints)), col = 4, xaxt="n")
lines(confint(tests, breaks = length(tests$breakpoints)), xaxt="n")
}
kable(break_points, caption = "Trend Structural breakpoints & dates")
```