GARCH Modelling

Importing libraries

Reading data

Important: Before running the code below, make sure your Knit directory is 'Document Directory'. This can be done by clicking the drop-down menu next to Knit, going to Knit directory and clicking on Document Directory.

```
setwd("..")
#Inputting log-returns
sp_logret<-read.csv("Data/Processed/sp_logret.csv")$x
dow_logret<-read.csv("Data/Processed/dow_logret.csv")$x
nas_logret<-read.csv("Data/Processed/nas_logret.csv")$x

#Inputting residuals
sp_residuals<-read.csv("Data/Processed/sp_residuals.csv")$x
dow_residuals<-read.csv("Data/Processed/dow_residuals.csv")$x
nas_residuals<-read.csv("Data/Processed/nas_residuals.csv")$x

#Calculating AR
sp_ar <- sp_logret - sp_residuals
dow_ar <- dow_logret - dow_residuals
nas_ar <- nas_logret - nas_residuals
```

Separating data into train and test:

```
sp_train_data_residuals <- sp_residuals[1:4000]
sp_test_data_residuals <- sp_residuals[4001:length(sp_residuals)]

dow_train_data_residuals <- dow_residuals[1:4000]
dow_test_data_residuals <- dow_residuals[4001:length(dow_residuals)]

nas_train_data_residuals <- nas_residuals[1:4000]
nas_test_data_residuals <- nas_residuals[4001:length(nas_residuals)]

sp_train_data_logret <- sp_logret[1:4000]
sp_test_data_logret <- sp_logret[4001:length(sp_logret)]

dow_train_data_logret <- dow_logret[1:4000]
dow_test_data_logret <- dow_logret[4001:length(dow_logret)]

nas_train_data_logret <- nas_logret[1:4000]
nas_test_data_logret <- nas_logret[4001:length(nas_logret)]
```

Choosing the optimal parameters of GARCH(p,q):

We will choose the GARCH(p,q) model which has the lowest AIC. We will also be using the Student's t-distribution to model our volatility.

We will first define a function which we will run on our residuals data for each index. The function will go through a series of GARCH models and output the one with the lowest AIC.

```
opm_garch <- function(x){
  final.aic <- Inf
  final.order <- c(0,0)
  for (i in 1:4) for (j in 1:4){
    x_model <-  ugarchspec(variance.model=list(garchOrder=c(i,j)),
                mean.model=list(armaOrder=c(0,0)), distribution.model = "std")
    current.aic <- infocriteria(ugarchfit(spec=x_model,data=x,solver='hybrid'))[1]
    if (current.aic < final.aic){
      final.aic <- current.aic
      final.order <- c(i,j)
  }
}
final.order
}
```

The below code is commented out as the algorithm takes some time to run and we don't need to run it as we already have obtained our previous parameters.

```
#opm_garch(sp_train_data_residuals)
#opm_garch(dow_train_data_residuals)
#opm_garch(nas_train_data_residuals)
```

The optimizer chose GARCH(2,1) to model the volatility for all of our indexes.

Hence, we have the following models for each index's log-returns:

- **sp:** AR(1)+GARCH(1,2)
- **dow:** AR(1)+GARCH(1,2)
- **nas:** ARMA(3,2)+GARCH(1,2)

We will now define our models for each index:

```
sp.model = ugarchspec(variance.model = list(model = 'sGARCH' , garchOrder = c(2,1)),mean.model = list(ar

dow.model = ugarchspec(variance.model = list(model = 'sGARCH' , garchOrder = c(2,1)),mean.model = list(a

nas.model = ugarchspec(variance.model = list(model = 'sGARCH' , garchOrder = c(2,1)),mean.model = list(a
```

Fitting our models:

Now we will fit our model on training data and output the coefficients:

```
sp.model.fit <- ugarchfit(sp.model, sp_residuals, solver = 'solnp', out.sample=1283)

dow.model.fit <- ugarchfit(dow.model, dow_residuals, solver = 'solnp', out.sample=1285)

nas.model.fit <- ugarchfit(nas.model, nas_residuals, solver = 'solnp', out.sample=1283)

#setwd('..')
#write.csv(sp.model.fit@fit[["sigma"]], 'Data/Processed/sp_sigma.csv', row.names=T)
#write.csv(dow.model.fit@fit[["sigma"]], 'Data/Processed/dow_sigma.csv', row.names=T)
#write.csv(nas.model.fit@fit[["sigma"]], 'Data/Processed/nas_sigma.csv', row.names=T)


options(scipen = 999)
sp.model.fit@fit$matcoef
```

```
##              Estimate    Std. Error   t value       Pr(>|t|)
## mu     0.000529206539 0.00013394819  3.950830 0.000077880604
## omega  0.000002544285 0.00000104574  2.432999 0.014974333785
## alpha1 0.034059467428 0.01394922171  2.441675 0.014619296537
## alpha2 0.088898037019 0.01938456245  4.586022 0.000004517702
## beta1  0.858681959616 0.01527802103 56.203742 0.000000000000
```

```
dow.model.fit@fit$matcoef
```

```
##              Estimate     Std. Error   t value      Pr(>|t|)
## mu     0.000534436366 0.000128295774  4.165658 0.00003104553
## omega  0.000002253363 0.000000975307  2.310414 0.02086526080
## alpha1 0.038558511056 0.015265327973  2.525888 0.01154062167
## alpha2 0.085217565476 0.019875520968  4.287564 0.00001806433
## beta1  0.858844273256 0.015467734468 55.524891 0.00000000000
```

```
nas.model.fit@fit$matcoef
```

```
##              Estimate     Std. Error   t value      Pr(>|t|)
## mu     0.000660959334 0.000169429859  3.901079 0.00009576473
## omega  0.000002735183 0.000001329919  2.056654 0.03971955214
## alpha1 0.021552650666 0.013685315916  1.574874 0.11528548984
## alpha2 0.083795133511 0.019627570327  4.269257 0.00001961256
## beta1  0.882204180364 0.015483720624 56.976240 0.00000000000
```

Our GARCH(p,q) models take the following shape:

S&P500:

$$\sigma_t^2 = 0.0000030 + 0.848\sigma_{t-1}^2 + 0.047\epsilon_{t-1}^2 + 0.079\epsilon_{t-2}^2$$

Dow Jones:

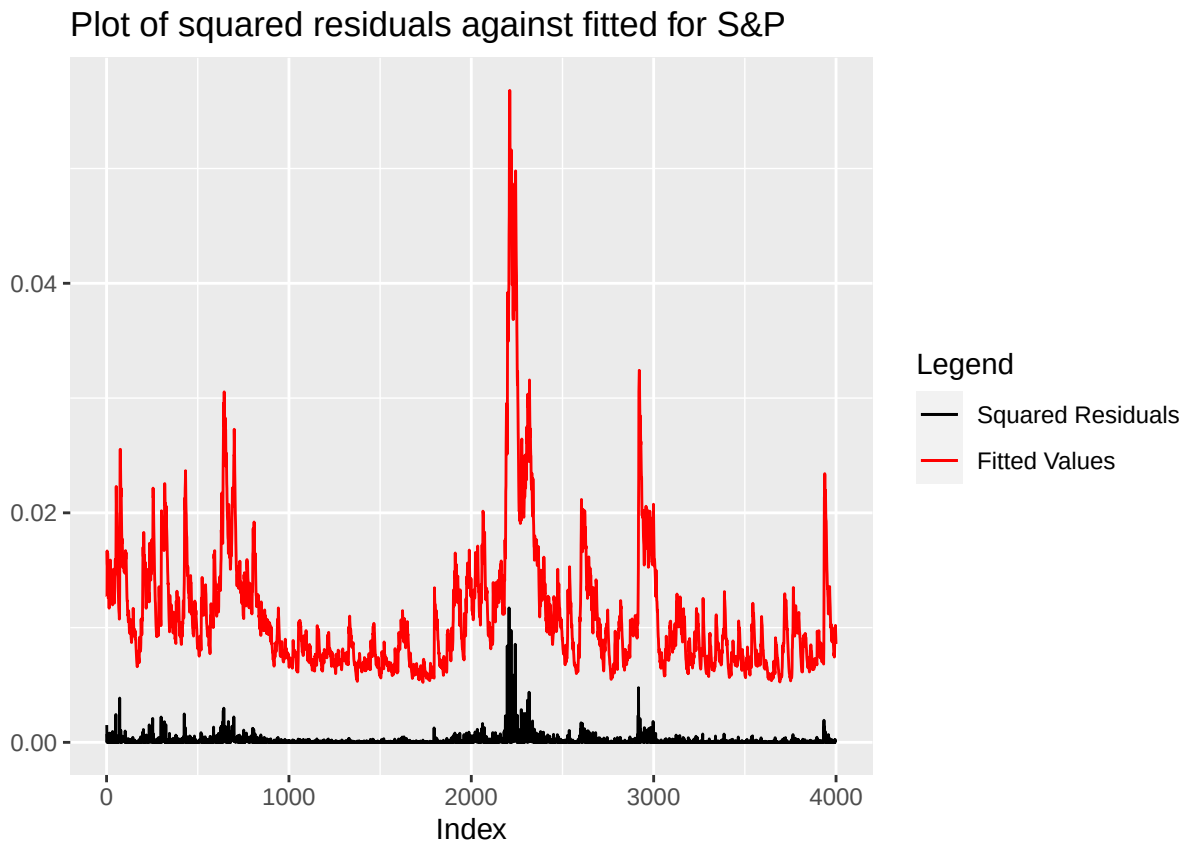
$$\sigma_t^2 = 0.0000022 + 0.859\sigma_{t-1}^2 + 0.039\epsilon_{t-1}^2 + 0.085\epsilon_{t-2}^2$$

NASDAQ:

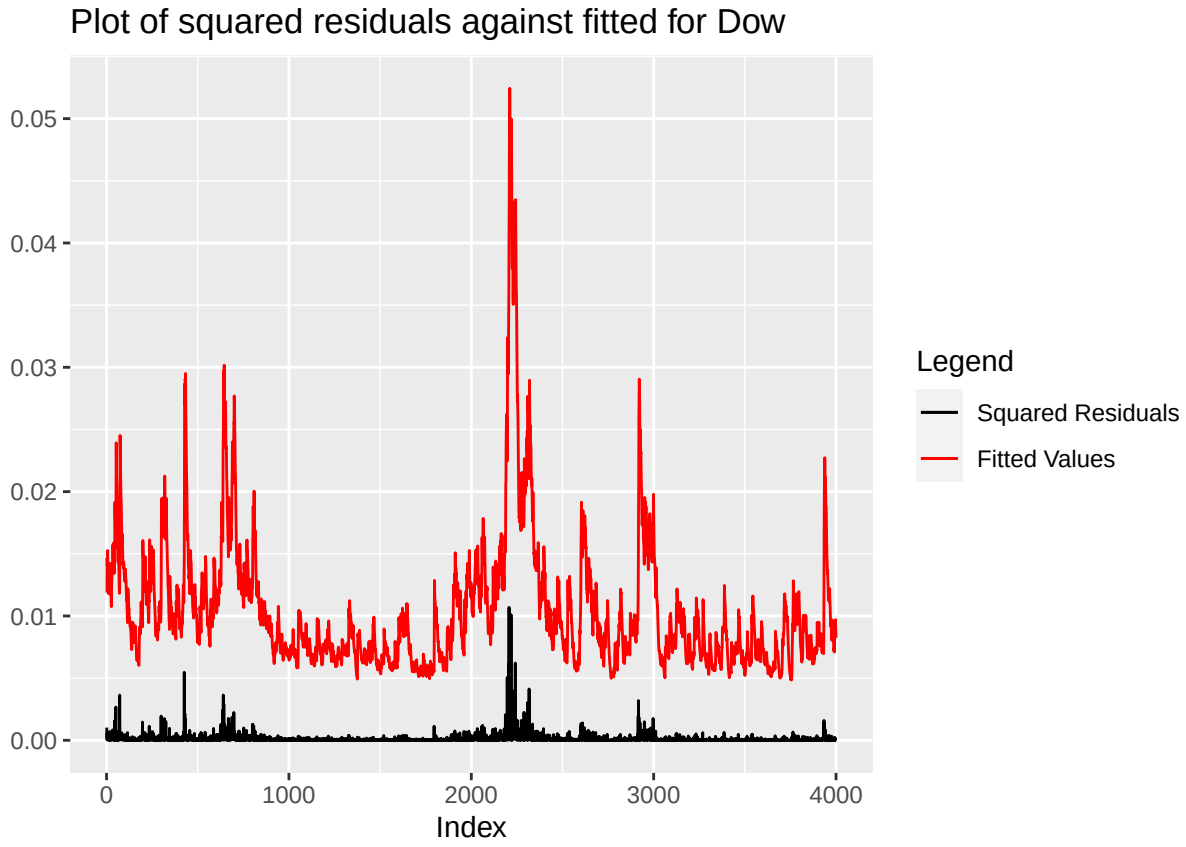
$$\sigma_t^2 = 0.0000027 + 0.882\sigma_{t-1}^2 + 0.022\epsilon_{t-1}^2 + 0.084\epsilon_{t-2}^2$$

Let us also produce some plots to compare our fitted values against the squared residuals:

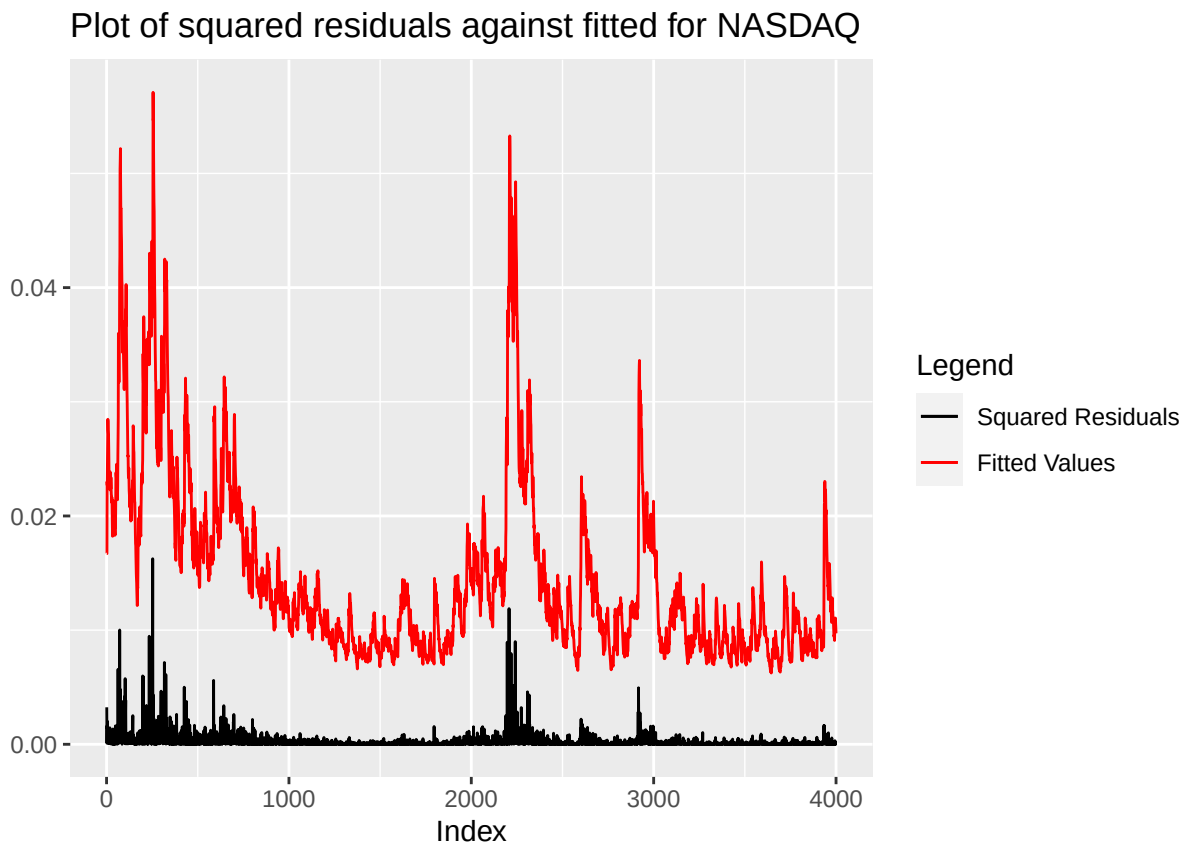
```
ggplot() +
  geom_line(aes(x=(1:length(sp_train_data_residuals)), y = (sp_train_data_residuals)^2, color = "Squared
  geom_line(aes(x=(1:length(sp_train_data_residuals)),y=sp.model.fit@fit[["sigma"]], color = "Fitted Val
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against fitted for S&P
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```




```
ggplot() +
  geom_line(aes(x=(1:length(dow_train_data_residuals)), y = (dow_train_data_residuals)^2, color = "Squa
  geom_line(aes(x=(1:length(dow_train_data_residuals)),y=dow.model.fit@fit[["sigma"]], color = "Fitted
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against fitted for Dow
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```




```
ggplot() +
  geom_line(aes(x=(1:length(nas_train_data_residuals)), y = (nas_train_data_residuals)^2, color = "Squa
  geom_line(aes(x=(1:length(nas_train_data_residuals)),y=nas.model.fit@fit[["sigma"]], color = "Fitted
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against fitted for NAS
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```



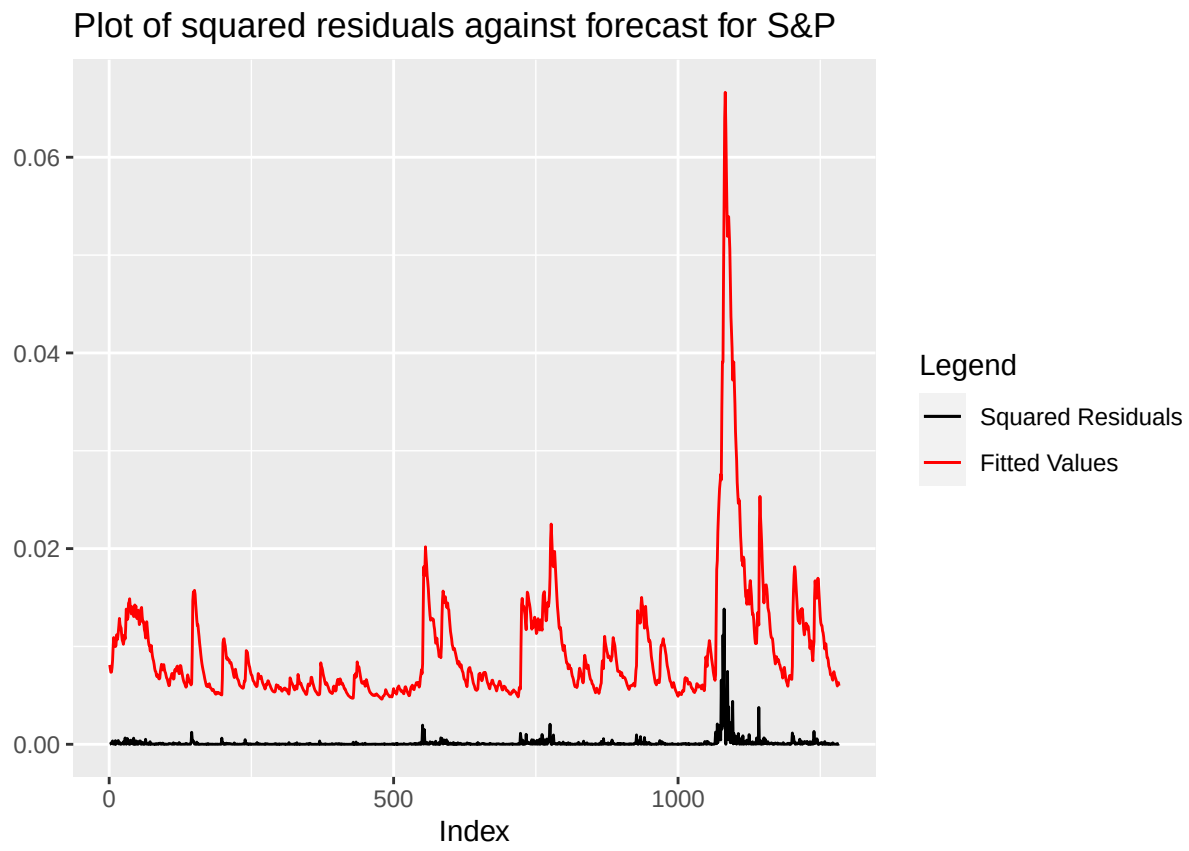
Forecasting on our model:

Now we will forecast on our data. The model forecasts one-day ahead and we will be refitting our model every 100 days:

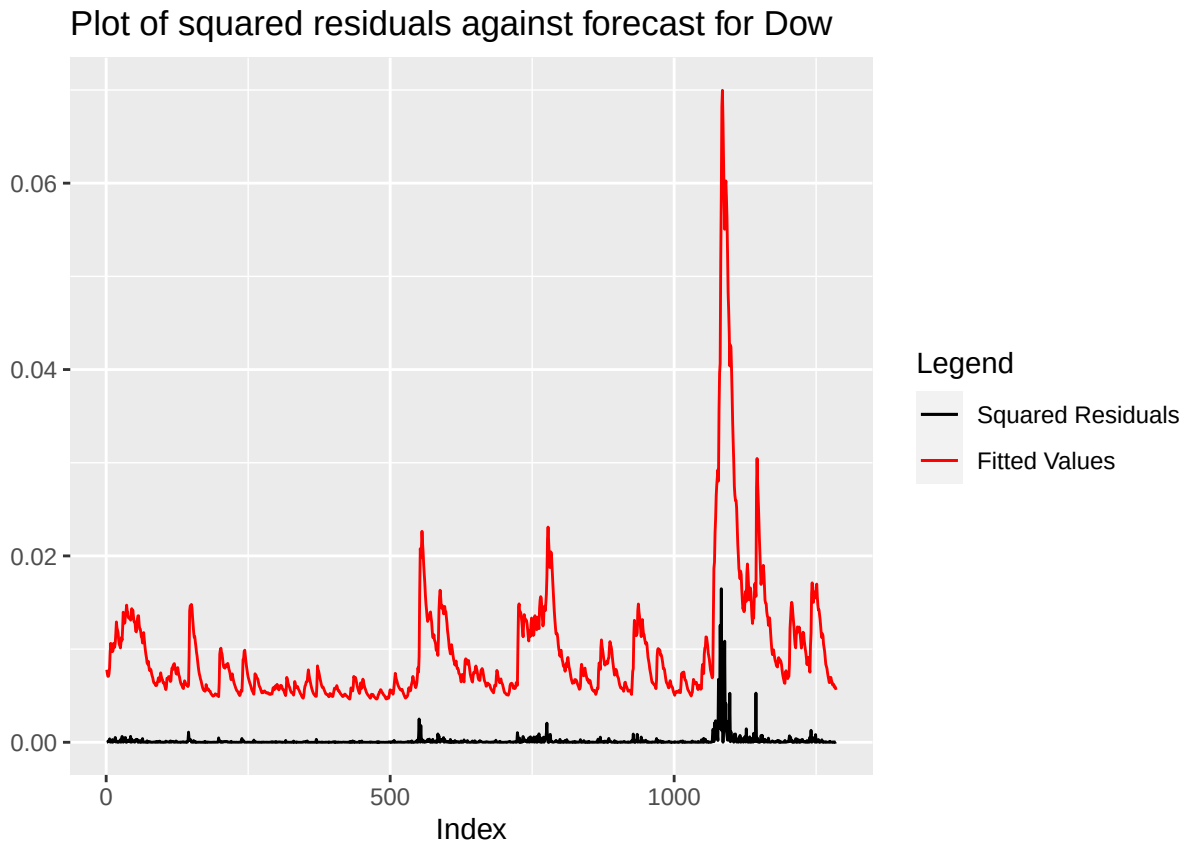
```
sp.model.forecast <- ugarchroll(spec = sp.model , data = sp_residuals, n.start = 4000, refit.every = 10

dow.model.forecast <- ugarchroll(spec = dow.model , data = dow_residuals, n.start = 4000, refit.every =

nas.model.forecast <- ugarchroll(spec = dow.model , data = nas_residuals , n.start = 4000, refit.every =
```

Let us produce some plots to check how our model does

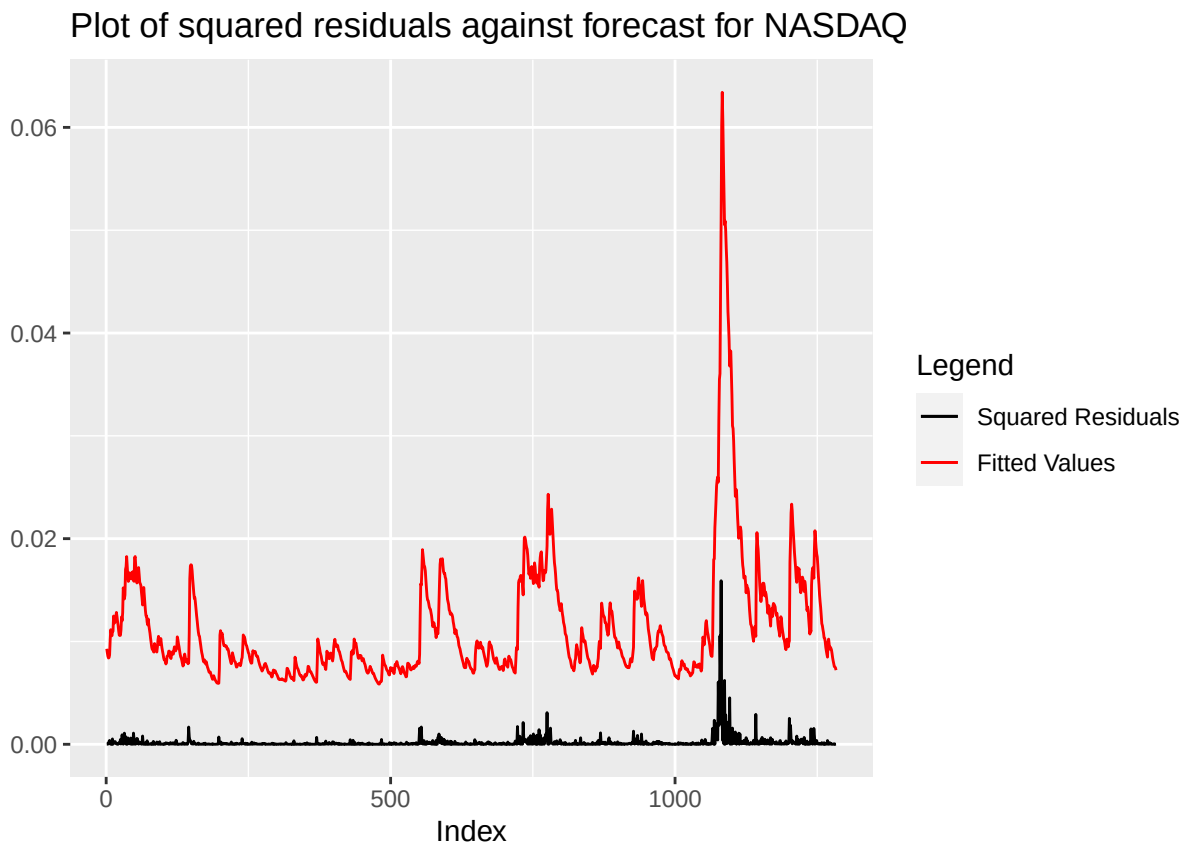
```
ggplot() +
  geom_line(aes(x=(1:length(sp_test_data_residuals)), y = (sp_test_data_residuals)^2, color = "Squared
  geom_line(aes(x=(1:length(sp_test_data_residuals)),y=sp.model.forecast@forecast$density[,'Sigma'], co
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against forecast for S
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```




```
ggplot() +
  geom_line(aes(x=(1:length(dow_test_data_residuals)), y = (dow_test_data_residuals)^2, color = "Squared
  geom_line(aes(x=(1:length(dow_test_data_residuals)),y=dow.model.forecast@forecast$density[,'Sigma'], c
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against forecast for Do
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```




```
ggplot() +
  geom_line(aes(x=(1:length(nas_test_data_residuals)), y = (nas_test_data_residuals)^2, color = "Squared
  geom_line(aes(x=(1:length(nas_test_data_residuals)),y=nas.model.forecast@forecast$density[,'Sigma'], c
  labs(x = "Index", y = "", color = "Legend", title = 'Plot of squared residuals against forecast for NA
  scale_colour_manual(values = c("Squared Residuals"="black", "Fitted Values"="red"))
```



Variance at Risk (Var):

In order to compare our LSTM with GARCH, we will calculate the VaR and see how well both models do.

Let's start of a with a quick definition of VaR:

'Value at risk (VaR) is a measure of the risk of loss for investments. It estimates how much a set of investments might lose (with a given probability), given normal market conditions, in a set time period such as a day. VaR is typically used by firms and regulators in the financial industry to gauge the amount of assets needed to cover possible losses.' (Wikipedia)

As seen before, we will model our returns according to the Student's t-distribution. Hence we will calculate VaR daily using the following formula:

$$VaR_{t|t-1}(\alpha) = \mu_t + \hat{\sigma}_{t|t-1} * F^{-1}(\alpha)$$

where $\hat{\sigma}_{t|t-1}$ is the conditional standard deviation given the information at $t-1$, F^-1(α) is the inverse PDF function of t-distribution at the significance level α and μ_t is the value of the mean equation at time t

In order to test our VaR estimates we will use a measure introduced in Kupiec (1995).

Let $N = \sum_{t=1}^{T} I_t$ where I_t is 1 if the actual log-return at time $t+1$ is greater than $VaR_{t+1|t}(\alpha)$ and 0 otherwise over T days. It was argued Kupiec (1995) that $N \sim Bin(T, alpha)$ i.e. N is binomially distributed. Hence we would expect N to be around $T * \alpha$.

Let us first define a function which we will use to calculate VaR:

```
VaR <- function(x.model,x_train_data_residuals,x_test_data_residuals,x_test_data_logret,x_ar,alpha){
  x_shape <- fitdist(distribution = 'std' , x = x_train_data_residuals)$pars[3]
  n = ceiling(length(x_test_data_residuals)/100)
  x_list <- rep(0,times=n)
  for (i in (1:n)){
    mean = na.omit(x_ar[4001:length(x_ar)][(100*(i-1)+1):(100*i)])
    sigma = na.omit(x.model@forecast$density[,'Sigma'][(100*(i-1)+1):(100*i)])
    VaR95_td <- mean+sigma*qdist(distribution='std', shape=x_shape, p=alpha)
    x_list[i] <- sum(na.omit(x_test_data_logret[(100*(i-1)+1):(100*i)]) < VaR95_td)
  }
  return(x_list)
}
```

We will break our test data into periods of 100 days and our significance level at 5%. Hence $T = 100$ and $\alpha = 0.05$. We will then take the average of all the periods. We expect the average to be around 5. The 95% confidence interval is $[1, 10]$ so we would expect N to lie within this range.

```
sp_list <- VaR(sp.model.forecast,sp_train_data_residuals,sp_test_data_residuals,sp_test_data_logret,sp_
dow_list <- VaR(dow.model.forecast,dow_train_data_residuals,dow_test_data_residuals,dow_test_data_logre
nas_list <- VaR(nas.model.forecast,nas_train_data_residuals,nas_test_data_residuals,nas_test_data_logre

mean(sp_list[1:12])
```

```
## [1] 7.666667
```

```
mean(dow_list[1:12])
```

```
## [1] 6.916667
```

```
mean(nas_list[1:12])
```

```
## [1] 8.583333
```

As we see, all our indexes lie within this range hence our GARCH models seems to be a good fit in modelling our VaR.

Expected Shortfall (ES):

We will now calculate the ES estimates of our model. Let us begin with a short introduction:

Expected shortfall (ES) is a risk measure—a concept used in the field of financial risk measurement to evaluate the market risk or credit risk of a portfolio. The "expected shortfall at q% level" is the expected return on the portfolio in the worst q% of cases. ES is an alternative to value at risk that is more sensitive to the shape of the tail of the loss distribution. (Wikipedia)

Practically, you can think of it as taking the average of the q% of cases.

Given that we know the VaR, we can calculate ES using the following equation:

$$\frac{1}{\alpha}\int_0^{\alpha} VaR(\alpha)\, d\alpha$$

We will use a similar method as VaR i.e. we split our test data into 100 days, calculate ES for every day and then average the 100 days.

```
VaRalpha <- function(x.model,x_train_data,x_test_data,x_ar,alpha,j,i){
        x_shape <- fitdist(distribution = 'std' , x = x_train_data)$pars[3]
        mean = x_ar[4001:length(x_ar)][(100*(i-1)+1):(100*i)]
        sigma = x.model@forecast$density[,'Sigma'][(100*(i-1)+1):(100*i)]
        VaR95_td <- mean[j]+sigma[j]*qdist(distribution='std', shape=x_shape, p=alpha)
}


ES <- function(x.model,x_train_data,x_test_data,x_ar,alpha){
  n = ceiling(length(x_test_data)/100)
  x_list <- rep(0,times=n)
  for (i in (1:n)){
    if (i==n){
      x_integrate <- rep(0,length(x_test_data)-(100)*(i-1))
      for (j in 1:(length(x_integrate))){
        x_integrate[j] <- as.numeric(integrate(VaRalpha,lower=0,upper=alpha,x.model=x.model,x_train_data
        x_list[i] <- mean(x_integrate)
      }
    }
    else {
      x_integrate <- rep(0,(100))
      for (j in 1:(length(x_integrate))){
        x_integrate[j] <- as.numeric(integrate(VaRalpha,lower=0,upper=alpha,x.model=x.model,x_train_data
        x_list[i] <- mean(x_integrate)
      }
    }
    print(i)
}
return(x_list)
}
```

```
setwd('..')

#sp_list_ES <- ES(sp.model.forecast,sp_train_data_residuals,sp_test_data_residuals,sp_ar,0.05)
#write.csv(sp_list_ES, 'Data/Processed/sp_list_ES.csv', row.names=T)

#dow_list_ES <- ES(dow.model.forecast,dow_train_data_residuals,dow_test_data_residuals,dow_ar,0.05)
#write.csv(dow_list_ES, 'Data/Processed/dow_list_ES.csv', row.names=T)

#nas_list_ES <- ES(nas.model.forecast,nas_train_data_residuals,nas_test_data_residuals,dow_ar,0.05)
#write.csv(nas_list_ES, 'Data/Processed/nas_list_ES.csv', row.names=T)

sp_list_ES<-read.csv("Data/Processed/sp_list_ES.csv")$x
dow_list_ES<-read.csv("Data/Processed/dow_list_ES.csv")$x
nas_list_ES<-read.csv("Data/Processed/nas_list_ES.csv")$x
```

We will compute the actual ES of the test data by averaging all the log-return prices which are less than 5% quantile.

```
ES_actual <- function(x_test_data,alpha){
  n <- ceiling(length(x_test_data)/100)
  x_list <- rep(0,times=n)
  for (i in (1:n)){
```

```
    if (i==n){
      x_data <- x_test_data[(100*(i-1)+1):length(x_test_data)]
      x_list[i] <- mean(x_data[x_data <= quantile(x_data,alpha)])
    }
    else {
      x_data <- x_test_data[(100*(i-1)+1):(100*i)]
      x_list[i] <- mean(x_data[x_data <= quantile(x_data,alpha)])
    }
  }
  return(x_list)
}
```

```
sp_list_ES_actual <- ES_actual(sp_test_data_logret,0.05)
dow_list_ES_actual <- ES_actual(dow_test_data_logret,0.05)
nas_list_ES_actual <- ES_actual(nas_test_data_logret,0.05)
```

We calculate the difference between the ES our model predicted and actual ES.

```
sp_ES_change <- mean((sp_list_ES_actual - sp_list_ES)^2)
dow_ES_change <- mean((dow_list_ES_actual - dow_list_ES)^2)
nas_ES_change <- mean((nas_list_ES_actual - nas_list_ES)^2)
```