The project is to model a stock return series of your choice with an ARMA + GARCH time series model. You will complete the project in two stages.

Part I: The first part focuses on the conditional mean. Find one or more candidate models for your data with the class of ARMA models.

Part II: The second part is to model your data with a full model, ARMA + GARCH, and complete the project.

Data Collection

The last stock return of your data should be 10/31/2024 or later. Check the requirements first.

Checking Data Requirements

Your data should be stock quotes from an individual company. Do NOT use any composite index, such as S&P 500, Dow Jones, or NASDAQ, etc. The stock needs to satisfy 3 conditions: (1) its IPO date was before January 1, 2009; (2) No missing values; (3) it has ARCH/GARCH property. Please follow the steps listed below.

Checking the IPO date. Load `quantmod` and `forecast` and down stock quotes. Please install if you have not.

```
> library("quantmod"); library("forecast")
> getSymbols("TICKER", from = "2009-01-01")
> head(TICKER, 2)
```

The starting date should be "2009-01-02". If it is later than "2009-01-02", select another stock.

Checking missing values. Use the following command.

```
> sum(is.na(TICKER))
```

If it returns 0, then there is no missing value. If it is greater than 0, please select another stock.

Checking ARCH effect. If the time series has ARCH/GARCH property, the squared observations are correlated.

```
> Yt = dailyReturn(Ad(TICKER), type = "log")[-1]
> Acf(Yt^2)
```

The significance of the first many ACFs should be very evident. If not, choose another stock.

Please do not include requirement checking as part of your project.

Preparing Data

After checking all 3 requirements, download the stock of your choice up to 11/01/2024 or later and prepare your data as follows.

Getting daily returns After downloading stock quotes, compute daily log returns (if `to = "2024-11-01"`).

```
> library(quantmod)
```

```
> getSymbols("TICKER", from = "2009-01-01", to = "2024-11-01")
> Yt = dailyReturn(Ad(TICKER), type = "log")[-1]
> colnames(Yt) = "TICKER"
> Yt = 100*Yt ## convert to %
> head(Yt, 2); tail(Yt, 2); dim(Yt)
```

The first date of `Yt` should be "2009-01-05", the last date should be "2024-10-31", the dimension should be 3984 by 1. If you choose the ending date later than `"2024-11-01"`, your dimension will be more than 3984 by 1.

Getting training set. You will use the first 11.5 years of data to model and leave the last 5.3 years of data for validation.

```
> ind = which(time(Yt) == "2020-06-30")
> Yn = Yt[1:ind,]; dim(Yn)
```

The dimension of `Yn` should be 2892 by 1. **Important:** Your sample the training set is `Yn`, the sample size is `n = dim(Yn)[1]` (which is 2892). Complete the modeling with `Yn`.

The remaining time series dated after "2020-06-30" is the validation set, which will be used to validate the model selected.

Save your data. E.g. Save as an R workspace file,

```
> save.image("/directory_of_your_choice/garch.RData")
```

Please note that the adjusted closing prices change almost every day, if you reload the stock quotes, you will get different set of numbers.

Notice that we did not remove the data class (xts) from our data (most of data I provided have the `xts` removed), some of the plots require our data to be in a time series class. Every time you reload your .RData file, you will need to load quantmod package by the R command library(quandmod).

Some Useful R Functions for ARMA Models

We mainly use R's time series functions. Some of functions in `forecast` package are equivalent with more options that can be useful. Always use `help(function)` to find out more information about any functions. My R functions are in 2 R files, `GARCH_RFunctions.R` and `GARCH_plotFunctions.R`.

Base R functions in `forecast` package Most of functions in the `forecast()` package are built on the base R functions but with more options. Here is the list of functions you may use.

R Base	Forecast	R Base	Forecast	R Base	Forecast
`arima()`	`Arima()`	acf(), pacf()	Acf(), Pacf()	predict()	forecast()

– `Arima()` & `arima()`: Fitting an ARMA model to data. `Arima()` gives both `aic` and `bic`, `arima()` returns only `aic`. Use them for fitting a specific model. Eg.

```
> Arima(Yn, order = c(2,0,1))    # fit an ARMA(2,1) to Yn
```

– `Acf()`, `Pacf()`, `acf()`, `pacf()`: Calculating autocorrelations and partial autocorrelations. The default setting has `plot = T`, plots will be generated. The major difference is that `acf()` plot from the 0th lag, `Acf()` does not include the 0th lag.

– `forecast()` & `predict()` Return h-step forecasts, `forecast()` also gives forecast intervals, `predict()` does not. Eg.

```
> fit = Arima(Yn, order = c(2,0,1))
> predict(fit, n.ahead = 10)   # 1-to-10-step ahead prediction
> forecast(fit, h = 10, level = c(68, 95))    # As above along with 68% and 95% PI's
```

Only in `forecast` package auto-selection function

– `auto.arima()`: Searches the "best" model with a model selection criterion, AIC, AICc or BIC. Eg.

```
> auto.arima(Yn, max.p = 3, max.q = 3, ic ="bic" )    # search by the BIC
```

Plot functions Most of plots can be produced by the generic plot function `plot()`. The `forecast` package also offers ggplots with generic function `autoplot()`.

– `plot()` for regular plots. Eg.

```
> plot(Yn) # Time series plot, find out more with help(plot.xts)
> fore21 = forecast(fit, h = 10, level = c(68, 95))
> plot(fore21) # plot time series with forecasts and PI being appended
```

– `autoplot()` of `forecast` packate for ggplots. Please note that ggplot does not take settings from `par()`.

```
> autoplot(Yn) # Time series plot of Yn
> autoplot(Acf(Yn, plot = F)) # or ggAcf(Yn), ACF plot of Yn
```

```
> autoplot(Pacf(Yn, plot = F)) # or ggPacf(Yn), PACF plot of Yn
> fore21 = forecast(fit, h = 10, level = c(68, 95))
> autoplot(fore21) # plot time series with forecasts and PI being appended
```

If you like to have more customized ggplot, you will need to install and upload the `ggplot2` package.

– Functions from my R file. These are for plotting AR and MA roots in the same complex plane.

```
> source("/path_you_saved/GARCH_plotFunctions.R")
> plot_roots(coef(fit)) # Regular plot of AR and MA roots in the same unit circle
> autoplot_roots(coef(fit)) # ggplot of AR and MA roots in the same unit circle
```

Some R functions for ARMA + GARCH Models

The R package `rugarch` is required for modeling ARMA + GARCH. Please see Handout 11 for details.

Functions from `rugarch` package R functions for ARMA + GARCH models.

– `ugarchspec()`: Specify an ARMA + GARCH to fit

– `ugarchfit()`: Fit the model specified by `ugarchspec()`

– `show()`: Show estimates, standard errors and tests of the fit from `ugarchfit()`.

– `ugarchforecast()`: Calculate forecasts and prediction errors with the model from `ugarchforecast()`.

For comparison purpose, refit an ARMA model with *i.i.d.* errors with the `ugarch`'s functions.

– `arfimaspec()`: Specify an ARMA model to be fit.

– `arfimafit()`: Fit and ARMA model specified with `arfimaspec()`.

– `arfimaforecast()`: Calculate forecasts and prediction errors with the model from `arfimafit()`.

R functions from my R file The functions are in GARCH_RFunctions.R. Please source the R file first,

```
> source("/path_you_saved/GARCH_RFunctions.R")
```

The following 2 R functions are for ARMA + GARCH model

– `showShort()`: Input the object from `ugarchfit()`. This is the same as `rugarch`'s `show()` function but without part of output that is not needed.

– `rate()`: Input the object from `ugarchforecast()`. It calculates the 90% and 95% coverage rates.

The following 2 R functions are for ARMA + *i.i.d.* errors

– `showShort0()`: Input the object from `arfimafit()` . This is the same as `rugarch`'s `show()` function but without part of output that is not needed.

– `rate0()`: Input the object from `arfimaforecast()`. It calculates the 90% and 95% coverage rates.

Plot functions from my `R` file he functions are in GARCH_plotFunctions.R. Please source the `R` file first,

```
> source("path_you_saved_the_file/GARCH_plotFunctions.R")
```

`plot_PI()` This function plots the rolling forecasts, the corresponding 95% PIs and outliers. Can be used to plot both ARMA + i.i.d. and ARMA + GARCH, similar to Figure 11.8 of Handout 11 but different. There are 2 must inputs, (1) rolling forecast of ARMA + i.i.d. Noise from `arfimaforecast()`, say `fore.0`; and (2) rolling forecast of ARMA + i.i.d. Noise from `ugarchforecast()`, say `fore.g`; then the plots can be obtained by `plot_PI(fore.0, fore.g)`. There is no ggplot version, but if you would like to have ggplot look-like plot set `gray.bg = T`, the default setting is `F`.

The plot required size specification to look properly in R Markdown as follows, e.g.

````
```{r, fig.width=7, fig.height=6}
plot_PI(fore.0, fore.g, gray.bg = T) # gray background
```
````

You should adjust the fig.width and fig.height to fit your requirement. All the line and point colors can be changed if you wish.

`plot_roots(), autoplot_roots()` These plots are used for ARMA fit before, they plot the roots of AR and MA polynomials. They can be used for the fit objects from `arfimafit()` and `ugarchfit()`. E.g. If `fig.g` is the fit from textttarfimafit(), the root plot can be obtained by

```
> plot_Roots(coef(fit.g))
```

Part I of project is to fit and ARMA model to your data `Yn` and retain a short list of candidate models for later use.

- Project should Include but not limit to data description, data exploring, candidate models, model selection, diagnostic checking and forecasts. This is not a sequence of steps, fitting a time series model usually with few refinements as discussed in class.
- The process of selecting a model is NOT a trial-and-error procedure. Avoid unnecessary steps, there is no advantage to be gained by making your project needlessly lengthy.
- Motivate and justify each step you are making and describe your findings in a way that shows your understanding of the material in this course.
- Do not produce any output or plot which you are not going to explain/describe. Every output and plot must be explained or interpreted.
- Both writing and coding will be evaluated. The attached rubric provides some information about how you would proceed with the project.

Part II of the project is complete fitting an ARMA + GARCH model to the data of your choice and validate your model.

- The ARMA part selected from Part I is not definite, it may require modification.
- Every point mentioned in the previous page apply. Always discuss your findings.
- Fit an ARMA + sGARCH model.
- Validate your model with rolling forecasts from Jan 2, 2018 to Oct 31, 2022 as described in the handouts. Compare them with the ARMA + *i.i.d.* white noise models. The distribution of the white noise needs to be carefully selected.
- * If the number of positive outliers and that of negative outliers are very different. Fit an extended GARCH model and do the same validation. We may not have time to discuss extended GARCH models.
- Compute the 1-day VaR for the investment $10000 of your stock based on your models, the ARMA + *i.i.d.* white noise models, the ARMA + sGARCH model and ARMA + □GARCH if there is one, assuming today is December 16, 2019, December 17, 2019 and December 18, 2019.
- Compute a consecutive five one-day VaR for the investment $10000 of your stock based on your models, the ARMA + *i.i.d.* white noise models, the ARMA + sGARCH model (and ARMA + □GARCH if there is one), for a low volatility period.

– Compute a consecutive five one-day VaR for the investment $10000 of your stock based on your models, the ARMA + *i.i.d.* white noise models, the ARMA + sGARCH model (and ARMA + □GARCH if there is one), for a high volatility period.

– The list gives only minimum requirements. You can include any analysis that is useful.

– Last but not least, the project is your final exam. It MUST be your own work to earn any credit.