Advanced Risk Management – Assignment 1

Deadline: 28 February 2022, 13.00h.

	Name	Student number	Email
1.			
2.			
3.			

Hand in the following via Canvas:

- Your notebook.
- A (printed) pdf version of your notebook. In Google Colab, this is most conveniently done in the Chrome browser, and then using the `File -> Print` menu option; you may have to print in landscape mode to make sure that everything appears in the pdf.
- Files must adhere to the following naming format to be considered eligible: 'Assignment1_STUDENTNUMBER1_STUDENTNUMBER2_STUDENTNUMBER3.ipynb' and 'Assignment1_STUDENTNUMBER1_STUDENTNUMBER2_STUDENTNUMBER3.pdf'. The files and the order of student numbers in the files names must be exactly identical for each member in a team.

Notes:

- The assignment is part of the examination, so the usual rules regarding plagiarism and fraud apply.
- Before submitting your work, click on `Runtime -> Restart and run all ...` and verify that your notebook produces the desired results and does not error.

Declaration of Originality: We, whose names are given under 1., 2., and 3., above declare that these solutions are solely our own work, and that we have not made these solutions available to any other student.

Introduction

The file `raw_data_2021.xlsx` contains daily data (January 2000 – January 2021, or a sub-period), for a number of international stock market indices, on the closing price and the daily realized variance RV (obtained from 5-minute returns). A list of the included indices is given on the website of the data provider, see https://realized.oxford-man.ox.ac.uk/data/assets. In this assignment, you are asked to estimate, test and compare several conditional volatility models for one of the indices in terms of their in-sample fit and their out-of-sample forecast quality.

Question 1: Load, clean, and display data

First, install and import the relevant libraries:

In [6]:
```
# !pip install arch             # uncomment for installing the arch package
import numpy as np
import os
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from arch import arch_model
from statsmodels.graphics.tsaplots import plot_acf
import seaborn as sns
```

Next, import the data and obtain the returns for all indices. Uncomment and adapt the lines necessary to mount the drive and change the path.

In [7]:
```
# from google.colab import drive
# drive.mount('/content/drive')
# path = '/content/drive/...'    # change path to your working directory
# os.chdir(path)

df = pd.read_excel('raw_data_2021.xlsx') # imports the data
```

Now we transform the data and put the data belonging to our favorite stock index in a DateFrame for further manipulation by running the code below. Pick your favorite index from the list of indices linked to above by adapting the 'Symbol' name below.

In [8]:
```
df['Date'] = df['Date'].str[0:10] # remove irrelevant parts of date string

df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d') # transform date strings to datetime
df = df.set_index(['Date']) # set index of DataFrame to the date column

df['R'] = np.log(df['close_price']) - np.log((df['close_price']).shift(1)) # calculate close-to-close returns
df.dropna() # drop N/A entries

sel = df['Symbol']=='.FCHI'   # Boolean array to select index;
                      # change '.ABC' to chosen index symbol, e.g., '.FCHI'

R = 100 * (df['R'].loc[sel])['2000-01-04':] # transform returns to percentages (recommended for GARCH package) and remove first row from returns

RV = 100**2 * (df['rv5'].loc[sel])['2000-01-04':] # transform realized variances as well
RV = RV / RV.mean() / (R**2).mean() # rescale realized variances to account for close-to-open news
RV = RV**0.5 # transform realized variance to realized volatility
```

Display a line graph of the returns, the realized volatility, and the standardizes returns (the return divided by its realized volatility). Calculate the skewness and kurtosis of the returns and the standardized returns. Summarize your findings. What can you conclude?

In [9]:
```
# #--- sample code
# series.skew() # returns the skewness of a series
# series.kurtosis() # returns the excess kurtosis of a series
# #---
```

Discussion of results: [10 pts.]

Question 2: Fitting a t distribution

It seems the returns have heavy tails. One way of modelling this property is to assume the returns are t distributed.

1. The code below fits a t ditribution to the data. Fit a t distribution to the data, assuming returns have mean zero. Provide the estimated value of the degrees of freedom parameter d and discuss your result.
2. Using the formula on Slide 16 of Week 3, estimate the pth quantile of the standardized t distribution with degrees of freedom parameter d. Use $p = 0.01$. Compare this value to the pth quantile of the standard normal distribution.
3. If we assume that the volatility is constant over time (an unrealistic assumption, but still one we can make), the quantiles can be used to obtain VaR forecasts for both distributions (see again Slide 16 of Week 3). Obtain the VaR forecasts and compute the percentage of exceedences. Use the standard deviation of the returns to estimate the constant volatility σ.

Discuss your findings for each step.

In [4]:
```
# #--- sample code
# d,location,scale = stats.t.fit(return,floc=0) # fits a t distribution to the series; the output params is a vector [d,location,scale]; floc=0 sets mean estimate to zero. We ONL
# pquantile = np.sqrt((d-2)/d) * stats.t.ppf(p,d,loc=0,scale=1) # returns the p-th quantile of the STANDARDIZED t distribution with degrees of freedom d, mean zero, and scale sd
# constantseries = 0 * series + c # returns a constantseries with same length as series and with every entry equal to c
# np.std(series) # returns the standard deviation of the series
# #---
```

Discussion of results [10 pts]:

Question 3: Extreme value theory

We can also use Extreme Value Theory to estimate VaR.

1. The simplest setting uses the formula for VaR on Slide 21 of Week 3: $\mathrm{VaR}^p_{t+1} = \sigma_{t+1} u \left(\frac{p}{T_u/T}\right)^{\xi}$, where u denotes the threshold, and T_u denotes the number of losses exceeding the threshold. Use $T_u = 0.05T$, being 5\% of the total data. Choose u equal to the 95% percentile of the losses. Estimate ξ with the Hill estimator. Describe which observations are used in this method (using the threshold). Also discuss what the estimate of ξ implies.

2. Set $\sigma_{t+1} = \sigma$, for all time periods t, and let σ denote the value found in Question 2. Obtain an estimate of VaR, at $p = 0.01$, and calculate the percentage of exceedences.

In [11]:
```
#--- sample code
# len(series) # obtain length of series
# int(np.floor(number)) # returns closest integer smaller than number (useful to make Tu an integer)
# series.quantile(n/100) # obtain nth percentile of a series
# # Hill estimator
# series_sorted = np.flipud(np.sort(series))  # sort series in descending order, largest values first
# xi_hill = np.mean(np.log(series_sorted[:Tu-1]/u)) # hill estimator (note: Tu must be an integer)
#---
```

Discussion of results [10 pts]:

Question 4: Adding time-varying volatility

In Question 1 we've found that the standardized returns display no excess kurtosis. The standardization was carried out using ex post realized volatility: We used the realized volatility observed at the end of the day t to standardize the return at the end of time t. To use the realized volatility for Value-at-Risk forecasting we must create ex ante forecasts of realized volatility.

1. We use a simple AR model to create Value-at-Risk forecasts. AR models can be estimated using OLS, by regressing the current value of RV on its lag and a constant (Use smf.ols() and don't forget to lag a regressor using d.shift(1).) Plot the ex post RV and its forecast together (Add the forecast to the DataFrame dfregress, dfregress['forecast'] = forecast, and use .plot()). Discuss your findings.

2. Create standardized return shocks (or simply shocks) by dividing the return by the volatility forecast from the AR(1) model and estimate the kurtosis.

3. Estimate VaR forecasts for the shocks using the standardized t distribution, standardized normal distribution and EVT theory. Don't forget to multiply the forecasts by the standard deviation of the shocks, in case of the standardized t and normal distribution. This is important, because the shocks have standard deviation quite far from 1. Then transform the VaR forecasts for shocks to VaR forecasts for returns by multiplying the VaR forecast for shocks by the realized volatility forecast. Plot the VaR forecasts for the returns together in a plot (You can use the DataFrame trick again). (Note: you must estimate the standard deviation of the shocks when you fit the t distribution, because these shocks might not have standard deviation close to one. The same holds for the normal distribution.)

4. Calculate the percentage of exceedences. Discuss your findings. Also relate your results to ealier questions. Would other tests be more helpful in finding differences between VaR forecasts based on constant volatility and time-varying volatility models?

In [2]:
```
# #--- sample code
# dfregress = pd.DataFrame({'Y': Y, 'X': X}) # create a Dataframe that contains X series (needed for smf.ols)
# model = smf.ols('Y ~ X.shift(1)', data=dfregress) # generate a linear regression model of X on the intercept and the lag of X
# # HINT: Think carefully about what Y and X should represent and whether you really need both in the AR(1) model
# result = model.fit()
# parametervector = np.array(result.params)
# forecast = np.array([X**0, X.shift(1)]).transpose().dot(parametervector)
# shock = series[1:]/forecast[1:] # if we standardize by a forecast we must remove the first value in each series
# #----
```

Discussion of results [15 pts]: