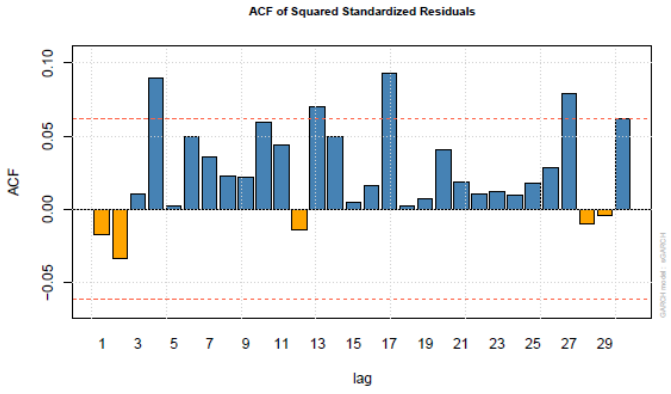
I. Introduction: Stock and industry brief description

NVIDIA Corporation Common Stock (ticker: NVDA) listed on Nasdaq is issued by Nvidia Corporation, which operates in the technology sector and is part of the semiconductor industry. As of October 2024, NVIDIA's stock price is around $130, with a $3.2 trillion market capitalization, making it one of the most widely traded stocks in the tech sector due to its dominance in GPUs and AI technology. According to Batra and others (2019), software ruled the industry for decades, with chips providing a necessary but underappreciated role. But things have changed with the emergence of machine learning and artificial intelligence (AI). The results from their analysis showed that by 2025, AI-related semiconductors might account for 20% of global demand and generate around $67 billion in revenue. This development significantly alters the role of semiconductor companies like NVIDIA inside the tech ecosystem and places them as key drivers of the future of technology. From 1 Sep 2018 to 31 Aug 2024 the stock shows an average positive return of 19%, indicating overall appreciation. However, the high variance (11.27) reflects significant volatility and frequent price swings. The slight negative skewness (-0.12) suggests larger potential losses than gains, while the kurtosis of 4.2 indicates a fat-tailed distribution, implying occasional extreme price movements.

II. Model description

Generally speaking, financial returns display the following characteristics: volatility clustering, fat tails, and volatility respond differently to positive and negative shocks. We will evaluate the adequacy of a model by evaluating whether the model can capture these characteristics.

First model: ARCH(2). The scope of this analysis restricts the mean equation to constant conditional mean, i.e. $r_t = \mu + \varepsilon_t;\ \varepsilon_t = \sigma_t z_t$ Hence, we first implement the model CCM-ARCH(2) of which the conditional variance can be modeled by: $\sigma_t^2 = w + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2$. After estimating the model using the returns from NVDA (Appendix 2), we obtain the fitted model ARCH(2): $r_t = 0.17 + \varepsilon_t;\ \sigma_t^2 = 6.48 + 0.21\varepsilon_{t-1}^2 + 0.25\varepsilon_{t-2}^2$, where $\varepsilon_t = \sigma_t z_t;\ z_t \sim N(0,1)$.



However, according to the ACF plot, the squared standardized residuals of ARCH(2) still display strong autocorrelation, hence ARCH(2) is not adequate to capture the volatility clustering.

Second model: GARCH(1,1). That requires another model with a more robust volatility clustering capturing effect, hence we now consider the GARCH(1,1) model: $r_t = \mu + \varepsilon_t$; $\varepsilon_t = \sigma_t z_t$; $\sigma_t^2 = w + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2$; $z_t \; i.i.d. \sim N(0,1)$. From the R results (Appendix 3), we get the fitted model as: $r_t = 0.18 + \varepsilon_t$; $\varepsilon_t = \sigma_t z_t$; $\sigma_t^2 = 0.44 + 0.13\varepsilon_{t-1}^2 + 0.84\sigma_{t-1}^2$; $z_t \; i.i.d. \sim N(0,1)$. This model assumes that positive and negative shocks have the same effects on volatility. However, in practice, there are asymmetric effects of shocks on conditional volatility. To be more specific, a negative shock in return tends to increase future volatility more than a positive shock of the same size. Moreover, the standardized residuals zt of this model still have heavy tail.

Third model: GJC.-GARCH(1,1). For those reasons, we now implement the GJC.-GARCH(1,1) model: $r_t = \mu + \varepsilon_t$; $\varepsilon_t = \sigma_t z_t$; $\sigma_t^2 = w + (\alpha_1 + \lambda_1\{\varepsilon_{t-i} < 0\})\varepsilon_{t-i}^2 + \beta_1 \sigma_{t-1}^2$; the standardized residual, z_t is assumed to follow a t-distribution with x degree of freedom z_t i.i.d ~ t(v). From the R output (Appendix 4), we can write down the fitted GJC-GARCH(1,1)-t(v) model as: $r_t = 0.17 + \varepsilon_t$; $\sigma_t^2 = 0.51 + (0.06 + 0.14\{\varepsilon_{t-i} < 0\})\varepsilon_{t-i}^2 + 0.83_1\sigma_{t-1}^2$ where $\varepsilon_t = \sigma_t z_t$ and z_t i.i.d ~ t(6.94).

To evaluate the in-sample fit, compare their AIC values—the lower the AIC, the better the model.

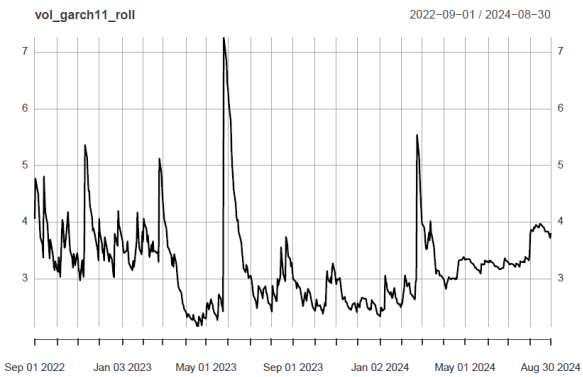
	ARCH(2)	GARCH(1,1)	GJR-GARCH(1,1)-t(v)
AIC	5.174	5.0959	5.0548

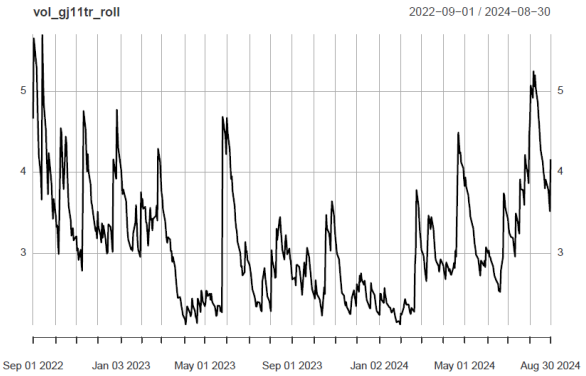
Based on the AIC Table obtained from R-output (Appendix 2, 3, 4), the models with the best in-sample fit, ranked in descending order, are GJR-GARCH(1,1)-t(v), GARCH(1,1), and ARCH(2).

III. Volatility analysis

1. One-step-ahead out of sample volatility forecast

We choose the two models with the best in-sample fit to conduct the one-step-ahead out-of-sample volatility forecast, those are GARCH(1,1) and GJC-GARCH(1,1)-t(6.94). The financial data, including NVIDIA's stock returns, often experience structural changes due to market regime shifts or economic policy changes, also known as parameter instability. Given the stock's high variance (11.27), which reflects significant volatility and frequent price swings, a rolling window continuously updates the model with recent data, enhancing adaptability to parameter instability and capturing evolving market dynamics effectively.

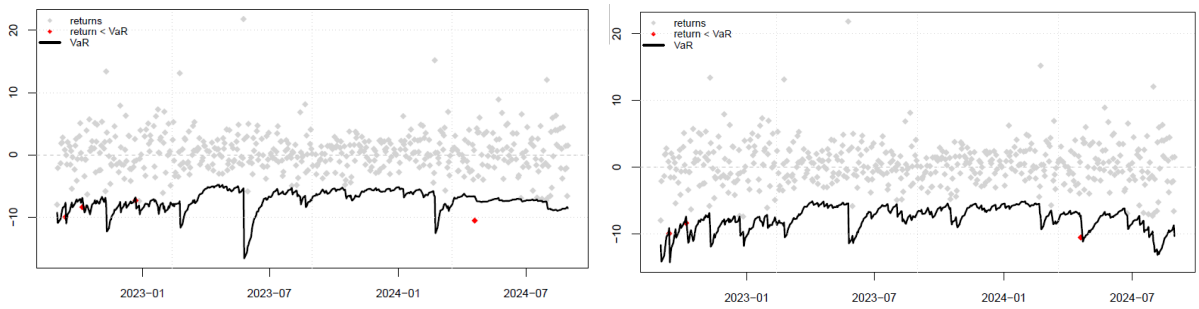




As seen in the GJR-GARCH volatility forecast plot, volatility spiked notably on September 1, 2022. This coincides with the sharp drop in U.S. chip stocks after Nvidia and AMD announced that U.S. officials had restricted exports of cutting-edge AI processors to China. Nvidia's stock plunged 11% on that day, driving volatility higher. Volatility was also elevated in May 2023, when Nvidia's stock was trading near $305 per share just before the company reported blowout earnings that sparked a surge in AI-related stocks, according to Morning Star

2. One-day 1% VaR of the stock in an out-of-sample period

One-day 1% VaR computed by GARCH(1,1) and GJR-GARCH(1,1)-t(6.94)



We conduct an Unconditional Coverage (UC) test and a Conditional Coverage (CC) test to backtest the VaR for both models. If our model for forecasting VaR is correctly specified, the probability of getting a hit using 1% VaR should approximate 1%. The number of hits should equal 1% of the sample size which is 1% of 502 approximately 5, and the exceedances should be evenly distributed across all time. Based on the R-output (Appendix 6), we reject the null hypothesis for the UC test and CC test for both models. For the UC test, rejecting the null hypothesis means that the actual number of exceedances is not significantly different from the expected value. And as the UC tests fail, the CC tests also fail. Overall, rejecting the null hypothesis in both tests suggests that both models (GARCH(1,1) and GJR-GARCH(1,1)-t(v)) do not accurately predict VaR in this context. The GARCH(1,1) model produces better VaR according to plot, as the exceedances are more evenly distributed across all time and the number of exceedances is closer to the expected number compared to that of GJR-GARCH(1,1)-t(v) (4 to 5 vs 3 to 5).

Conclusion

NVIDIA Corporation (NVDA) is a key player in the semiconductor industry, benefiting from its dominance in AI and GPU technologies. Between September 2018 and August 2024, NVIDIA's stock exhibited significant volatility, with a high variance (11.27) and occasional extreme price swings. We applied ARCH(2), GARCH(1,1), and GJR-GARCH(1,1)-t(v) models to analyze volatility, with the GJR-GARCH(1,1)-t(v) providing the best in-sample fit based on AIC values. However, neither the GARCH(1,1) nor the GJR-GARCH(1,1)-t(v) accurately predicted Value at Risk (VaR) during the out-of-sample period, as evidenced by the failure of both Unconditional and Conditional Coverage tests. Nonetheless, the GARCH(1,1) model showed relatively better performance in terms of exceedances and distribution of risk.

APPENDIX

1.

```
> basicStats(returns)
             NVDA.Adjusted
nobs           1507.000000
NAs               0.000000
Minimum         -20.771162
Maximum          21.808788
1. Quartile      -1.599364
3. Quartile       2.003667
Mean              0.187931
Median            0.288373
Sum             283.211685
SE Mean           0.086489
LCL Mean          0.018279
UCL Mean          0.357583
Variance         11.272910
Stdev             3.357515
Skewness         -0.124171
Kurtosis          4.196966
>
```

2.

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

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

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu       0.16965    0.092004   1.8439 0.065193
omega    6.48309    0.552055  11.7435 0.000000
alpha1   0.21225    0.057120   3.7158 0.000203
alpha2   0.25214    0.054881   4.5943 0.000004

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu       0.16965    0.106018   1.6002 0.109556
omega    6.48309    1.061896   6.1052 0.000000
alpha1   0.21225    0.069691   3.0456 0.002322
alpha2   0.25214    0.082860   3.0430 0.002343

LogLikelihood : -2595.956

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

Akaike       5.1740
Bayes        5.1936
Shibata      5.1740
Hannan-Quinn 5.1815
```

3.

```
Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu       0.18341    0.087328   2.1003 0.035703
omega    0.43853    0.142511   3.0771 0.002090
alpha1   0.12832    0.026263   4.8860 0.000001
beta1    0.83760    0.031324  26.7394 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu       0.18341    0.090148   2.0346 0.041892
omega    0.43853    0.171652   2.5547 0.010627
alpha1   0.12832    0.031430   4.0828 0.000044
beta1    0.83760    0.039247  21.3418 0.000000

LogLikelihood : -2556.712

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

Akaike       5.0959
Bayes        5.1155
Shibata      5.0959
Hannan-Quinn 5.1034
```

4.

```
Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.171368    0.083727   2.0467 0.040683
omega   0.514539    0.188157   2.7346 0.006245
alpha1  0.063571    0.029650   2.1441 0.032026
beta1   0.825706    0.036799  22.4380 0.000000
gamma1  0.140069    0.054834   2.5544 0.010637
shape   6.947542    1.477679   4.7017 0.000003

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.171368    0.084159   2.0362 0.041726
omega   0.514539    0.184949   2.7821 0.005402
alpha1  0.063571    0.030110   2.1113 0.034749
beta1   0.825706    0.039675  20.8117 0.000000
gamma1  0.140069    0.052136   2.6866 0.007218
shape   6.947542    1.567720   4.4316 0.000009

LogLikelihood : -2534.059

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

Akaike       5.0548
Bayes        5.0842
Shibata      5.0548
Hannan-Quinn 5.0660
```

5.

```
> VaRTest(alpha=0.01,realizedret,var1_garch11)
$expected.exceed
[1] 5

$actual.exceed
[1] 4

$uc.H0
[1] "Correct Exceedances"

$uc.LRstat
[1] 0.2250074

$uc.critical
[1] 3.841459

$uc.LRp
[1] 0.6352507

$uc.Decision
[1] "Fail to Reject H0"

$cc.H0
[1] "Correct Exceedances & Independent"

$cc.LRstat
[1] 0.2893944

$cc.critical
[1] 5.991465

$cc.LRp
[1] 0.8652842

$cc.Decision
[1] "Fail to Reject H0"
```

```
> VaRTest(alpha=0.01,realizedret,var1_gjr11t)
$expected.exceed
[1] 5

$actual.exceed
[1] 3

$uc.H0
[1] "Correct Exceedances"

$uc.LRstat
[1] 0.9592934

$uc.critical
[1] 3.841459

$uc.LRp
[1] 0.327365

$uc.Decision
[1] "Fail to Reject H0"

$cc.H0
[1] "Correct Exceedances & Independent"

$cc.LRstat
[1] 0.9954382

$cc.critical
[1] 5.991465

$cc.LRp
[1] 0.6079157

$cc.Decision
[1] "Fail to Reject H0"
```

6. R Code

```
# Clean the memory
rm(list = ls())

# Clean the screen
cat("\014")   # Equivalent to "ctrl+L"

# Set the working directory
setwd("C:/aMonash/Quant Fin/Assignment 2")

# Load required library
library(quantmod)
library(fBasics)
library(jtools)
library(stats)
library(forecast)
library(rugarch)

###1. Data collection
# compute returns
getSymbols("NVDA",from="2018-09-01",to="2024-08-31",periodicity = 'daily')
logprices <- log(NVDA$NVDA.Adjusted)
returns <- diff(logprices)*100
returns <- na.omit(returns)
```

```
head(returns)
tail(returns)

basicStats(returns)

###2. Risk Analysis:

# define a new xts object using only data until 31 Aug 2022
returns_is <- returns["/2022-08-31"]
head(returns_is)
tail(returns_is)

# Get all data from 1 Sep 2018 to 31 Aug 2022 as in-sample data for estimation
T1 <- length(returns_is)
T2 <- length(returns["2022-09-01/"])

# specify the 1st model we want to use to estimate the returns
spec_arch2 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(2,0))
                  mean.model = list(armaOrder = c(0, 0)),
                  distribution.model = "norm")
# fit the model
arch2 = ugarchfit(spec = spec_arch2, data = returns,out.sample = T2)
show(arch2)
```

```
# choose 11 for acf of squared standardized residuals (z^2)
plot(arch2)
11
0
dev.copy(pdf,'Figures/squared_residual_arch2.pdf', width = 8, height = 5,paper='special')
dev.off()

# Specify the 2nd model we want to use to estimate the returns
# GARCH(1,1)
# specify the model we want to use to estimate the returns
spec_garch11 = ugarchspec(variance.model = list(model="sGARCH",garchOrder = c(1,1)),
                  mean.model = list(armaOrder = c(0,0)),
                  distribution.model="norm")
# fit the model
garch11 = ugarchfit(spec=spec_garch11,data=returns,out.sample=T2)
show(garch11)

# choose 11 for acf of squared standardized residuals (z^2)
plot(garch11)
11
0
dev.copy(pdf,'Figures/squared_residual_garch11.pdf', width = 8, height = 5,paper='special')
dev.off()

# Specify the 3rd model we want to use to estimate the returns
# GJR-GARCH(1,1)-t(v)
spec_gjr11t = ugarchspec(variance.model = list(model = "gjrGARCH", garchOrder = c(1,1)),
                         mean.model = list(armaOrder = c(0, 0)),
                         distribution.model = "std")
# fit the model
gjr11t = ugarchfit(spec = spec_gjr11t, data = returns,out.sample = T2)
show(gjr11t)
```

```
### Conduct one-step-ahead out-of-sample volatility forecasts using two models

##### out-of-sample comparison
dates_oos = index(returns[(T1+1):(T1+T2)]) # specify the out-of-sample dates

### rolling window estimation for first model
roll_garch11 = ugarchroll(spec_garch11, returns, n.start = T1, refit.every = 10,
                          refit.window = "moving")
# retrieve one-day-ahead volatility forecast
vol_garch11_roll = roll_garch11@forecast$density$Sigma
# redefine an xts object to store volatility forecasts
vol_garch11_roll = xts(vol_garch11_roll,order.by = dates_oos)

plot(vol_garch11_roll)
dev.copy(pdf,'Figures/vol_garch11_roll.pdf', width = 8, height = 5,paper='special')
dev.off()

### rolling window estimation for second model
roll_gjr11t = ugarchroll(spec_gjr11t, returns, n.start = T1, refit.every = 10,
                         refit.window = "moving")
# retrieve one-day-ahead volatility forecast
vol_gjr11t_roll = roll_gjr11t@forecast$density$Sigma
# redefine an xts object to store volatility forecasts
vol_gj11tr_roll = xts(vol_gjr11t_roll,order.by = dates_oos)

plot(vol_gj11tr_roll)
dev.copy(pdf,'Figures/vol_gj11tr_roll.pdf', width = 8, height = 5,paper='special')
dev.off()

# save the realized returns in the out-of-sample period to use as a proxy
realizedret = returns[(T1+1):(T1+T2)]
```

```
# 1% Value-at-risk (VaR) & Back-testing
str(roll_garch11@forecast)
# retrieve one-day-ahead 1% VaR forecast for First Model
var1_garch11 = roll_garch11@forecast$VaR$`alpha(1%)`
  # redefine an xts object to store volatility forecasts
  var1_garch11 = xts(var1_garch11,order.by = dates_oos)

  # plot the VaR and the realized return
  VaRplot(alpha=0.01,realizedret,var1_garch11)
  dev.copy(pdf,'Figures/VaRplot_garch11.pdf', width = 8, height = 5,paper='special')
  dev.off()

# retrieve one-day-ahead 1% VaR forecast for Second Model
var1_gjr11t = roll_gjr11t@forecast$VaR$`alpha(1%)`
  # redefine an xts object to store volatility forecasts
  var1_gjr11t = xts(var1_gjr11t,order.by = dates_oos)

# save the realized returns in the out-of-sample period
realizedret = returns[(T1+1):(T1+T2)]

  # plot the VaR and the realized return
  VaRplot(alpha=0.01,realizedret,var1_gjr11t)
  dev.copy(pdf,'Figures/VaRplot_gjr11t.pdf', width = 8, height = 5,paper='special')
  dev.off()

# evaluating the two VaR forecasts
VaRTest(alpha=0.01,realizedret,var1_garch11)
VaRTest(alpha=0.01,realizedret,var1_gjr11t)
```

REFERENCE

1. Niero, B. NVIDIA's AI Revolution: A Comprehensive Strategic Analysis.
2. Batra, G., Jacobson, Z., Madhav, S., Queirolo, A., & Santhanam, N. (2019). Artificial-intelligence hardware: New opportunities for semiconductor companies. McKinsey and Company, 2.
3. Jadagoudar, A. (2024). Growth Trajectories: A Comparative Analysis of Nvidia Corporation and Cisco Systems in Technological Advancements and Market Dynamics. *Available at SSRN 4786491*.