

Functional structure of excess return and volatility

Author: Chenxi Zhao

Advisor: Marcos Escobar-Anel

Abstract

Capturing the relation between excess returns and volatility can help making better decisions in the stock market in terms of portfolio allocation and assets risk management. This paper takes the data of a minute-by-minute series of S&P500 from January 2009 to January 2021 as the research object and explores the best structural representation for the excess return as a function of the volatility, for a well-known index. This is implemented via regression models for volatility and excess returns. The results reveal that there's a structural break in the relationship between the excess return and volatility based on the sign of the excess return, the functional connection could be either linear, logarithmic or quadratic.

1. Introduction

Businesses, investors and consumers are mostly likely reluctant to make spending and investment decisions in a market with high uncertainty. The tradeoff between risk and return has long been an important topic in asset valuation research. There is widespread agreement that, over a given time period, investors demand a higher expected return from a riskier security, but this desire might not be supported by the performance of stocks as indicated by data. Moreover, there is no agreement about the relation between risk and return across time.

The empirical literature on this topic has attempted to characterize the nature of the linear relation between the excess return and volatility. However, the reported findings are conflicting. For example, Campbell and Hentschel (1992) and French, Schwert, and Stambaugh (1987) conclude that the data are consistent with a positive relation between conditional expected excess return and conditional variance, whereas Fama and Schwert (1977), Campbell (1987), Pagan and Hong (1991), Breen, Glosten, and Jagannathan (1989), Turner, Startz, and Nelson (1989), and Nelson (1991) find a negative relation in several datasets. Chan, Karolyi, and Stulz (1992) find no significant variance effect for the United States, but implicitly find one on the world market portfolio. Harvey (1989) provides empirical evidence suggesting that there may be some time variation in the relation between risk and return over time.

In order to find the best structural representation for the volatility and the excess return of a well-known index, we selected the data of S&P500 from January 9, 2009, to January 22, 2021, for experiments and comparisons, and do analysis of regression models for excess return as the dependent variable and volatility as the independent variable. During the calculation, we noticed that the free rate of interest is very small, resulting in little difference between excess return and expected return, so that we can directly explore the relationship between the expected return and volatility. Based on the experimental results, we found that:

When we do regressions including the whole period under analysis, no matter how we transform the variables, there is no significant relationship between the expected return and volatility. However, if we do linear regressions by splitting the data into two parts according to the positive and negative sign of the expected returns, there is a logarithmic (linear and quadratic are also acceptable) relationship between the expected return and volatility.

The remainder of this report proceeds as follows. The second section is dedicated to the methodology of data calculation and regression analysis used in the experiment. This is then followed by the numerical results of two types of regression analysis. Section 4 presents the results and discusses the main findings of our study, and a conclusion then brings the paper to an end.

2. Methodology

In this section, we explain how the functional data analysis methodology can be applied to our problem.

Notation

$S_{t,m}$ – Stock price of day t . m is time in minutes. We assume, for convenience, M minutes per day (usually less than 480, and different in different days), $m=1, \dots, M$.

$r_{f,y}$ – Risk-free rate, y is time in years, $y=1, \dots, Y$.

r_m – Rate of return per minute, m is time in minutes.

μ_d – Daily expected return, d is time in days, $d = 1, \dots, D$.

σ_d – Daily volatility, d is time in days.

r_d – Daily return, d is time in days.

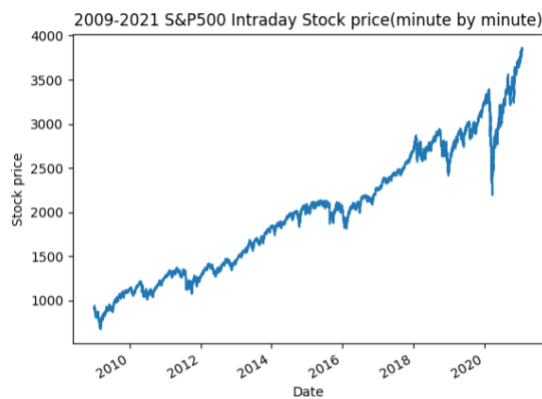
μ_y – Annualized expected return, y is time in years.

σ_y – Annualized volatility, y is time in years.

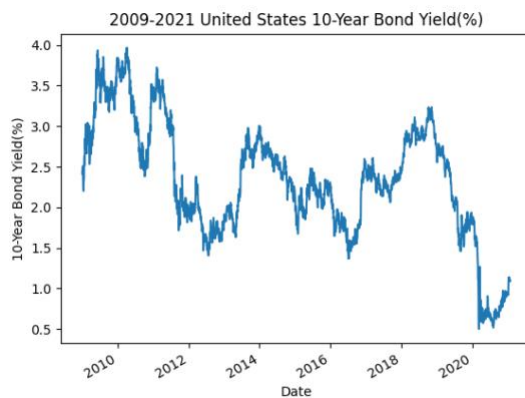
p_y – Excess return, y is time in years.

Data Sources:

- (1) S&P500 intraday price from 2009-01-09 to 2021-01-22 (minute by minute)



- (2) United States 10-year bond yield from 2009-01-09 to 2021-01-22



2.1 Data Cleaning and Calculation

Suppose the excess return of stock k is known as the expected return minus the risk-free rate, where we use the U.S. 10-year bond yield rate as $r_{f,y}$.

First, given daily time series of SP500, we use the minute-by-minute index data to calculate the rate of return per minute r_m , which can be expressed as

$$r_m = \frac{S_{t,m+1}}{S_{t,m}} - 1$$

Because the time available consists of irregular time intervals, we exclude the first minute in the calculations.

Second, we calculate the daily expected return and daily volatility. Assume the average return (per minute) is 0.01%, if we compound \$1 per minute, then the investor should have $\$(1 + 0.0001)^M$ by the end of the day. By the Taylor expansion,

$$(1 + 0.0001)^M = 1 + \ln(1.0001) M + O^2 \approx 1 + 0.0001M + O^2$$

Due to the average return per minute is small, the remainder O^2 would be super small, we could approximate $(1 + 0.0001)^M$ by $1 + 0.0001M$. Therefore, we are calculating the daily expected return by multiplying M , where M equals 480 as we assume an eight-hour working day system. For the volatility, we multiply by the square root of the time scaling factor. The formulas can be expressed as

$$\mu_d = 480 * E(r_m)$$

$$\sigma_d = \sqrt{480 * Var(r_m)}$$

For the purpose of precision, we discuss another way of calculating the rate of return. We use the open price of day $t+1$ and day t to calculate the daily return r_d , which can be expressed as

$$r_d = \frac{S_{t+1,1}}{S_{t,1}} - 1$$

This might be a good proxy for the daily expected return $\mu_{k,d}$.

For consistency, we convert the rates from daily to annually, where we could carry out the assumption we made when converting from minutes to days. We make the definition that the annual expected return equals the number of workdays in a year multiplies the daily expected return. To convert daily volatility to annual volatility, multiply by the square root of the number days in a year, which can be expressed as

$$\mu_y = 250 * \mu_d$$

$$\sigma_y = \sqrt{250} * \sigma_d$$

where we assume a year has 250 workdays.

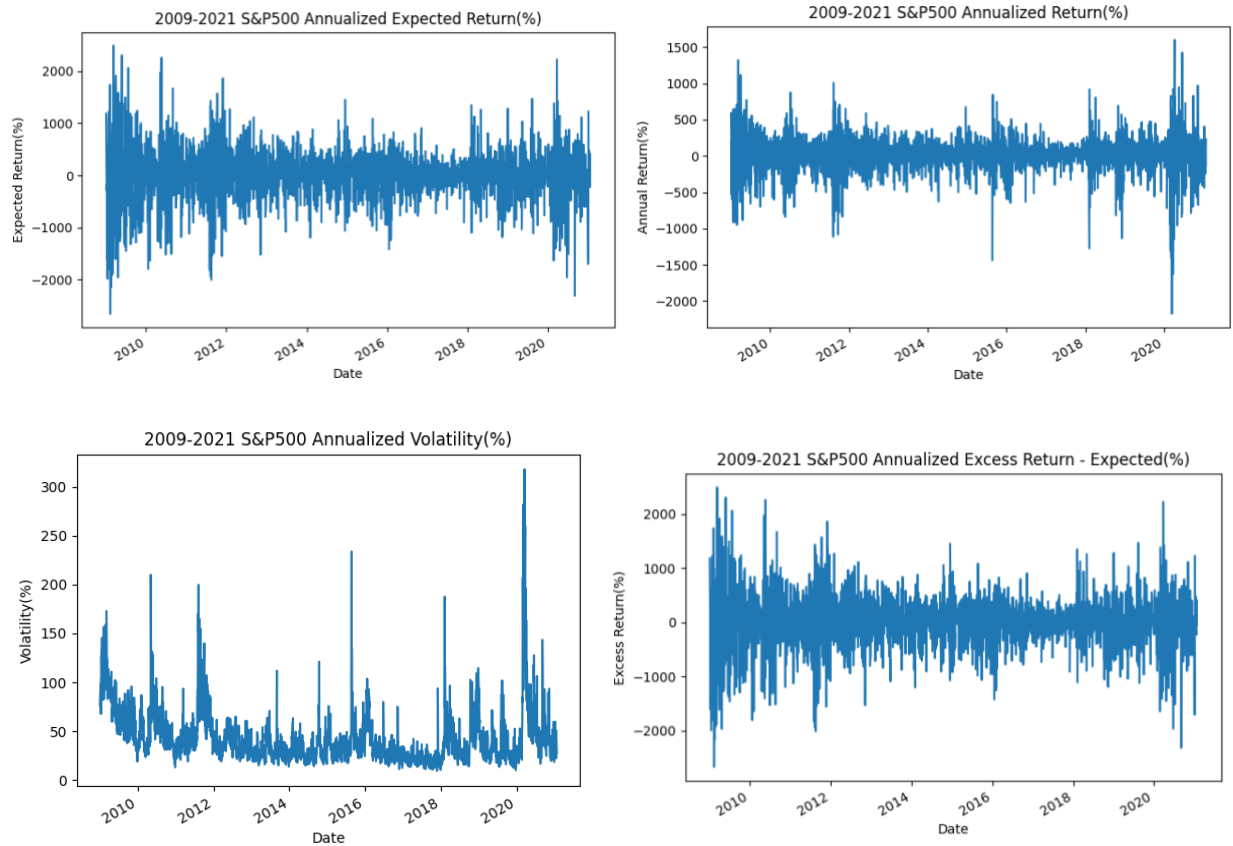
Now we get the annual excess return as $p_y = \mu_y - r_y$.

Note: Issues might arise here when converting from daily to annually.

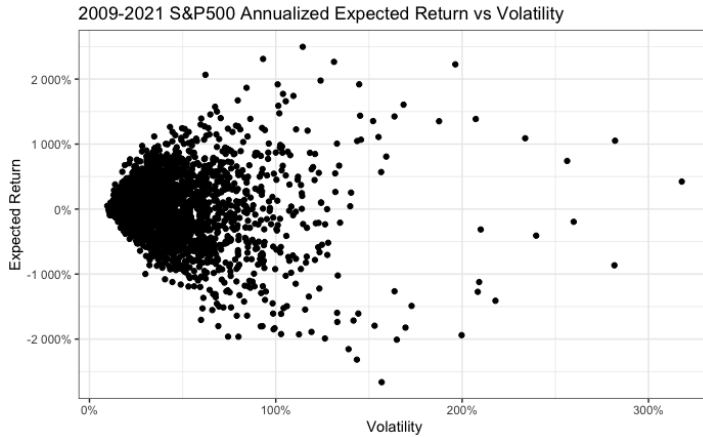
1. The approximation of multiplication instead of compounding is not precise for large daily expected return μ_d . The large returns make the remainder O^2 large as well, causing large calculation errors. For example, assume the daily expected return is 1%, if we compound per day, then annual expected return should be $(1 + 0.01)^{250} - 1$, which is 1103.22%. If we calculate by multiplication, the return would be 250%. As you can see, there is a huge difference. The multiplication underestimates the rate of return.
2. We use the square-root rule for the time-scaling of volatility. This method of scaling volatility is only appropriate if returns are independently and identically distributed. However, daily (or high frequency) stock prices are generally not independently and identically distributed. In this case Diebold et al. (1997) shows that scaling volatility to a longer time horizon can amplify fluctuations in the data; scaled volatilities are often overestimated.

2.2 Visualization of calculation results

By using pandas, NumPy and other tools within Python (See Appendix – Codes), we calculated our target data and visualized them as follows:



Our goal is to explore the best structural representation for the volatility and excess return. The above data calculation shows that the interest rate is so small that it could be ignored for now, so we could directly explore the relationship between the volatility and expected return using linear regressions.



2.3 Regressions

As the scatter plot shows above, the points are crowded due to the large amount of data. To facilitate our exploration of the relationship between the expected return and volatility, we first select one of the years of data for regression analysis (i.e. whenever possible we use linear regression, in some cases we simply do a least squares minimization in the spirit of non-linear regression). Here we choose the year 2014 to start with as the market is more stable.

When assessing the regression models, we use mean squared error (MSE) as a way to measure the amount of error, which can be calculated as:

$$MSE = \frac{\sum(y_t - \hat{y}_t)^2}{T}$$

where:

y_t is the annualized expected return on day t

\hat{y}_t is the corresponding predicted value on day t

T is the number of observation days

A model with a smaller MSE is preferred when deciding the best structural representation.

Our analysis is performed with outliers removed. We use Cook's Distance for regression outlier detection. Cook's distance, often denoted D_i , is used in regression analysis to identify influential data points that may negatively affect the regression model. The formula for Cook's distance is:

$$D_i = \left(\frac{r_i^2}{p} * MSE \right) * \frac{h_{ii}}{(1 - h_{ii})^2}$$

where:

r_i is the i th residual

p is the number of coefficients in the regression model

MSE is the mean squared error

h_{ii} is the i th leverage value

We follow the general rule that any point with a Cook's Distance over $4/n$ (where n is the total number of data points) is considered to be an outlier. (See Appendix – Codes)

3. Numerical Results

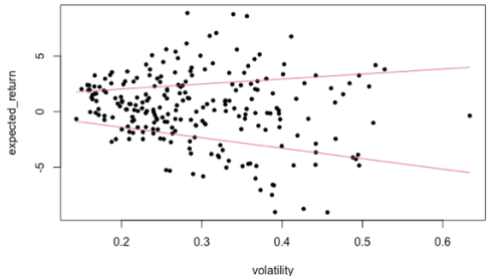
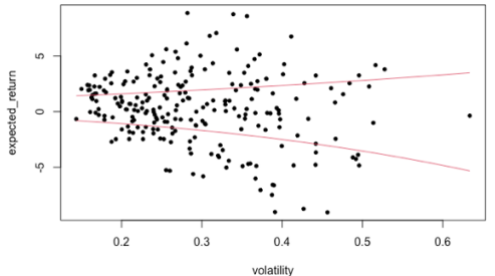
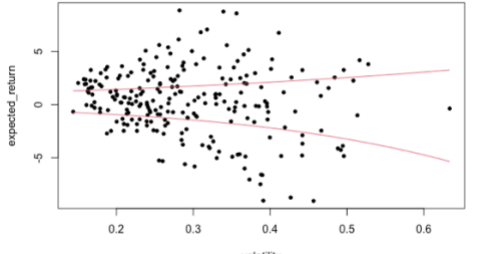
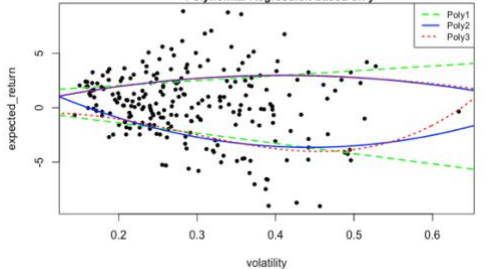
Here we conduct two types of analyses, one works with the whole data set, fitting a model accordingly (type 1), the second analysis (type 2) splits the data in two parts, positive and negative returns, fitting the two pieces accordingly.

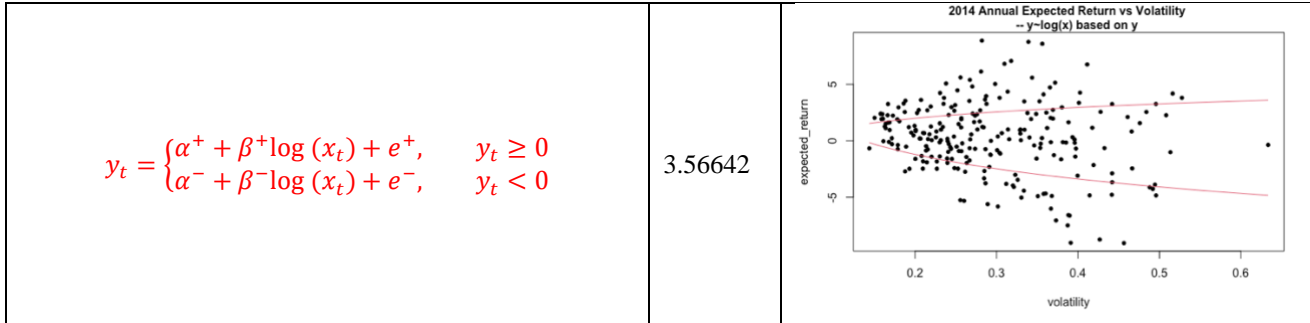
3.1 Analysis of type 1

Linear regression model		MSE	Graphs
-	y_t : annualized expected return on day t		
-	x_t : annualized standard deviation on day t		
Outliers included	$y_t = \alpha + \beta x_t + e$	13.20523	
Outliers removed	$y_t = \alpha + \beta x_t + e$	9.540857	
	$y_t = \alpha + \beta_1 x_t + \beta_2 x_t^2 + e$	9.531777	
	$y_t = \alpha + \beta_1 x_t + \beta_2 x_t^2 + \beta_3 x_t^3 + e$	9.49771	
	$\sqrt[3]{y_t} = \alpha + \beta x_t + e$	9.878104	
	$\sqrt[3]{y_t} = \alpha + \beta \sqrt[3]{x_t} + e$	9.866972	/
	$y_t = \alpha + \beta \log(x_t) + e$	9.539443	/
	$\exp(y_t) = \alpha + \beta x_t + e$	29.3166	/
	$y_t = \alpha + \beta \frac{1}{x_t} + e$	9.559679	/
	$y_t = \alpha + \beta \arcsin(\sqrt{x_t}) + e$	9.539842	/
	$\operatorname{sech}(y_t) = \alpha + \beta x_t + e$	11.9276	/
$\operatorname{arcsinh}(y_t) = \alpha + \beta \operatorname{arcsinh}(x_t) + e$	9.620362	/	
$\tanh(y_t) = \alpha + \beta \tanh(x_t) + e$	9.72895	/	

We perform transformations on the variables with the expectation that the linear regression model could work for that transformed data. However, our attempts didn't bring significant improvements on the MSE compared with the simple linear regression. Surprisingly, when performing a cubic root transformation on y_t , we noticed two linear patterns on the graph, which brings up the idea that we could do two linear regressions by splitting the expected return into positive and negative.

3.2 Analysis of type 2

Linear regression model (based on y_t)	MSE	Graphs
$p\sqrt[y_t]{y_t} = \begin{cases} \alpha^+ + \beta^+ x_t + e^+, & y_t \geq 0 \\ \alpha^- + \beta^- x_t + e^-, & y_t < 0 \end{cases}$	$p = 1$ 3.62911	
	$p = 3$ 4.00278	
	$p = 9$ 4.31824	
$y_t = \begin{cases} \alpha^+ + \beta_1^+ x_t + \beta_2^+ x_t^2 + e^+, & y_t \geq 0 \\ \alpha^- + \beta_1^- x_t + \beta_2^- x_t^2 + e^-, & y_t < 0 \end{cases}$	3.47818	
$y_t = \begin{cases} \alpha^+ + \beta_1^+ x_t + \beta_2^+ x_t^2 + \beta_3^+ x_t^3 + e^+, & y_t \geq 0 \\ \alpha^- + \beta_1^- x_t + \beta_2^- x_t^2 + \beta_3^- x_t^3 + e^-, & y_t < 0 \end{cases}$	3.43619	

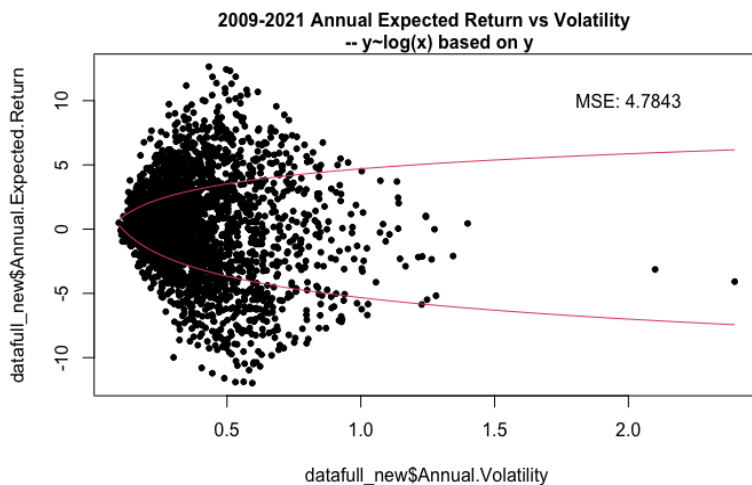


4. Results

Doing two linear regressions based on the discussion of classifying the expected return led to a significant reduction in MSE, which made the model fit our data much better. Considering the financial standpoint, we choose $y_t = \begin{cases} \alpha^+ + \beta^+ \log(x_t) + e^+, & y_t \geq 0 \\ \alpha^- + \beta^- \log(x_t) + e^-, & y_t < 0 \end{cases}$ with MSE equals 3.56642 as our best structural representation for the volatility and expected return, which indicates a logarithmic relationship between the expected return and volatility.

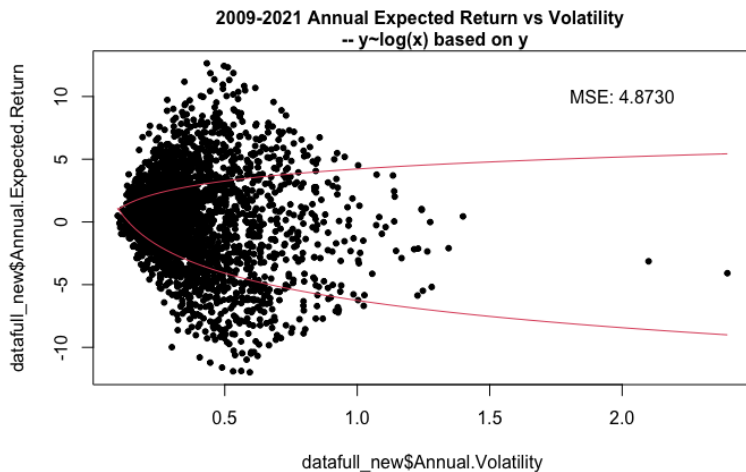
Then, we repeat this structure to all years, as well as a model with all years at once.

MSE of all years (Using $y_t = \begin{cases} \alpha^+ + \beta^+ \log(x_t) + e^+, & y_t \geq 0 \\ \alpha^- + \beta^- \log(x_t) + e^-, & y_t < 0 \end{cases}$)												
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
MSE	19.05943	8.83983	8.92778	4.52497	3.2786	3.56642	3.9409	3.0367	1.04024	4.7549	3.9474	8.69023
α^+	5.8190	5.7657	5.5013	3.5255	4.1187	4.2203	2.8044	4.0038	2.5975	3.9663	5.0886	5.1511
β^+	1.1713	2.6809	1.7990	0.6548	1.2780	1.3849	0.4387	1.2323	0.6756	1.3537	1.9590	1.9633
α^-	-7.1393	-8.4895	-6.1769	-6.9150	-4.5799	-6.2574	-7.5638	-4.8907	-5.3249	-6.2119	-5.4877	-8.2834
β^-	-4.2120	-6.4191	-3.1190	-3.9759	-1.8483	-3.1376	-4.1805	-2.1111	-2.4549	-3.2274	-2.5284	-5.3642



$$y_t = \begin{cases} 4.6942 + 1.6904 \log(x_t) + e^+, & y_t \geq 0 \\ -5.3327 - 2.4102 \log(x_t) + e^-, & y_t < 0 \end{cases}$$

MSE of all years (Using the model of 2014 : $y_t = \begin{cases} 4.2203 + 1.3849 \log(x_t) + e^+, & y_t \geq 0 \\ -6.2574 - 3.1376 \log(x_t) + e^-, & y_t < 0 \end{cases}$)											
2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
20.73622	9.54769	9.41422	4.58464	3.3269	3.56642	4.1349	3.11657	1.1819	4.79027	4.0027	9.41198



$$y_t = \begin{cases} 4.2203 + 1.3849 \log(x_t) + e^+, & y_t \geq 0 \\ -6.2574 - 3.1376 \log(x_t) + e^-, & y_t < 0 \end{cases}$$

From the tables above, we observed that directly fitting the other years data into the model of 2014 only brings a slightly higher MSE compared with applying the logarithmic structure to all years, which suggests that the model of 2014 is a good indicator to reality.

There are four relatively high MSE in table. Considering that 2009-2011 was affected by the aftermath of the 2008 financial crisis and 2020 was influenced by the Coronavirus pandemic, we inferred that these abnormal MSEs are reasonable as the financial market is volatile.

According to the regression analysis, we tend to conclude that, in the case of classifying the expected return into positive and negative, there is a logarithmic relationship between the expected return and volatility.

Future Research

Our work can be continued and improved in many directions. For instance, we would have explored the statistical differences in parameters between the various years as a way of confirming the model's parameters could change in time, requiring further analysis. We could also explore other indexes and stocks to see if our findings of a fitting split between positive and negative returns is supported across the market. We could have also compared the three leading models, i.e. linear, quadratic and logarithmic, more closely to see if there is a clear winner beyond the MSE measure. These all comments and many more non-disclosed shall be part of future research.

References

- Campbell, John Y. and Ludger Hentschel (1992). No news is good news: An asymmetric model of changing volatility in stock returns, *Journal of Financial Economics* 31, 281–318.
- French, Kenneth R., G. William Schwert, and Robert F. Stambaugh (1987). Expected stock returns and volatility, *Journal of Financial Economics* 19, 3–29.
- Fama, Eugene F., and G. William Schwert (1977). Asset returns and inflation, *Journal of Financial Economics* 5, 115–146.
- Campbell, John Y. (1987). Stock returns and the term structure, *Journal of Financial Economics* 18, 373–399.
- Pagan, Adrian R., and Y. S. Hong (1991). Nonparametric estimation and the risk premium, in William Barnett, James Powell, and George Tauchen, Eds.: *Nonparametric and Semiparametric Methods in Econometrics and Statistics* (Cambridge University Press, Cambridge), pp. 51–75.
- Breen, William, Lawrence R. Glosten, and Ravi Jagannathan (1989). Economic significance of predictable variations in stock index returns, *Journal of Finance* 44, 1177–1189.
- Nelson, Daniel B. (1991). Conditional heteroskedasticity in asset returns: A new approach, *Econometrica* 59, 347–370.
- Chan, K. C., G. A. Karolyi, Rene M. Stulz (1992). Global financial markets and the risk premium on U.S. equity, *Journal of Financial Economics* 32, 137–167.
- Harvey, Campbell R., 1989, Time-varying conditional covariances in tests of asset pricing models, *Journal of Financial Economics* 24, 289–317.
- Christoffersen, P. F., & Diebold, F. X. (1997). How Relevant is Volatility Forecasting for Financial Risk Management? Manuscript, Research Department, International Monetary Fund, and Department of Economics, University of Pennsylvania.

Appendix – Codes

Data Cleaning and Calculation (Python)

```
import time
import os
import pickle as pk
import pandas as pd
import numpy as np

DailyStd="Volatility"
DailyReturn="Daily Expected Return" # 480*mean(Minutely return)
ReturnPerMinute="Minute Return"
ExcessReturn="Excess Return"

def get_day_minutereturn(df):
    tic=time.time()
    row=df.shape[0]
    df.loc[:, "Minute Return"] = np.nan
    for i in range(row):
        # Minutely return
        ts = df.loc[i, "Date"] # Exclude the first minute type(ts):str
        if ts[-4:] == "9:30":
            continue
        sp500_cur = df.loc[i, "SP500"]
        sp500_pre = df.loc[i - 1, "SP500"]
        df.loc[i, "Minute Return"] = (sp500_cur / sp500_pre) - 1
    return df

def get_day_day_count(df):
    # Calculate how many days
    day, day_count = [], []
    count=1
    row=df.shape[0]
    day.append(df.loc[0, "Date"].split(" ")[0])

    for i in range(df.shape[0]):
        ts=df.loc[i,"Date"]
        ts2day=ts.split(" ")[0]
        if ts2day in day:
            count+=1
        else:
            day.append(ts2day)
            day_count.append(count)
            count=1
        if i==row-1:
            day_count.append(count)
    # for n in range(len(day)):
    #     if n==0:
    #         continue
    #     day_count[n]=day_count[n]+1
    return day, day_count

def get_daily_return_volatility(df, day_count_dict):
    # daily_return=480*mean(minute_return)
    # daily_std=sqrt(480)*std(minute_return)
    tic=time.time()
    row=df.shape[0]
    i=0
    daily_return=[]
    daily_std=[]
    while i<row:
        sum_num_today=day_count_dict[df.loc[i, "Date"].split(" ")[0]]
        index=i+sum_num_today
        # print(np.nanmean(df.loc[i:index-1, "Minute Return"]))
        today_return=480*np.nanmean(df.loc[i:index-1, "Minute Return"])
        daily_return.append(today_return)
        today_std=np.sqrt(480)*np.nanstd(df.loc[i:index-1, "Minute Return"])
        daily_std.append(today_std)
        i+=sum_num_today
    time_used=time.time()-tic
```

```

return daily_return, daily_std

def get_excess_return(df):
    # df=data2
    tic=time.time()
    row=df.shape[0]
    df.loc[:, "Excess Return"]=0
    for i in range(row):
        # print(df.loc[i, "Adj Close"])
        if df.loc[i, "Adj Close"]!=0:
            df.loc[i, "Excess Return"]=df.loc[i, DailyReturn]
        else:
            df.loc[i, "Excess Return"]=df.loc[i, DailyReturn]-((df.loc[i, "Adj
Close"]/100+1)**(1/365)-1)
    return df

## Program start ##
# Loading datas
if "data1.csv" in os.listdir("."):
    data1=pd.read_csv("data1.csv")
else:
    data1=pd.read_csv("EURUSD_DAX_SPX_final.csv")

print(list(data1.columns.values))
if "Minute Return" not in list(data1.columns.values):
    data1=get_day_minutereturn(df=data1)
    data1.to_csv("data1.csv", index=None, encoding="utf-8")
if "day.pk" not in os.listdir(".") or "day_count.pk" not in os.listdir("."):
    day, day_count = get_day_day_count(df=data1)
    day_save="day.pk"
    day_count_save="day count.pk"
    with open(day_save, "wb") as f1:
        pk.dump(day, f1)
        f1.close()
    with open(day_count_save, "wb") as f2:
        pk.dump(day_count, f2)
        f2.close()
else:
    day_opened=open("day.pk", "rb")
    day_count_opened=open("day_count.pk", "rb")
    day=pk.load(day_opened)
    day_count=pk.load(day_count_opened)
    day_opened.close()
    day_count_opened.close()
    day_count[0]=day_count[0]-1

day_count_dict={}
for i in range(len(day)):
    day_count_dict[day[i]] = day_count[i]
#for key, value in day_count_dict.items():
    #if value<136:
        #print(key, value)
daily_return, daily_std=get_daily_return_volatility(df=data1, day_count_dict=day_count_dict)

if "data2.csv" in os.listdir("."):
    data2=pd.read_csv("data2.csv")
else:
    data2 = pd.read_excel("Interest_rate-10yBond.xls")

if ExcessReturn not in list(data2.columns.values):
    day_num=len(day)
    origin_data_num=data2.shape[0]
    data2.drop(columns="Date", inplace=True)
    tmp_df=pd.DataFrame(data=[0]*(day_num-origin_data_num), columns=["Adj Close"])
    data2=data2.append(tmp_df, ignore_index=True)
    data2.loc[:, "Date"] = day
    data2.loc[:, DailyReturn] = daily_return # Add the calculated daily expected return to data2
    data2.loc[:, DailyStd] = daily_std # Add the calculated volatility to data2
    data2_excess_return = get_excess_return(df=data2)
    data2.to_csv("data2.csv", index=None, encoding="utf-8")

```

```

data2_excess_return = get_excess_return(df=data2) # Add the calculated excess return to data2
data2.to_csv("data2.csv",index=None,encoding="utf-8")

# Calculate daily return
import pandas as pd
import datetime
df = pd.read_csv('EURUSD_DAX_SPX_final.csv')
df['date'] = df['Date'].astype(str).str[0:10]
df['Time'] = df['Date'].astype(str).str[11:19]
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d %H:%M')
a = df.groupby('date').apply(lambda x: x.loc[x['Date'].idxmin(), 'SP500']).to_frame()
b = df.groupby('date').apply(lambda x: x.loc[x['Date'].idxmin(), 'SP500']).to_frame()
b.reset_index(inplace=True)
a['index'] = range(len(a))
a['index'] = a['index'] + 1
a.reset_index(inplace=True)
df_new = pd.merge(a, b, left_on='index', right_on='index', how='left')
df_new['value'] = df_new['0_y'] / df_new['0_x'] - 1
df_new[['date_x', 'value']].to_csv('a.csv')

```

Linear Regression (R)

```

#2014
data2014 <- read.csv('2014.csv')
fit <- lm(volatility~expected_return, data2014)
plot(expected_return~volatility,data=data2014,pch=20,
      main = "2014 Annual Expected Return vs Volatility
            --Simple linear regression",cex.main=1)
abline(fit,col=2,lwd=2)
text(x=1, y=12, paste("y = 0.3152 - 0.0034x", "\nMSE:13.20523"))
mean((data2014$expected_return-fitted(fit))^2)

#Remove influential points
influ_idx1 = which(cooks.distance(fit) > 4 / length(cooks.distance(fit)))
influ_idx1
length(influ_idx1)
sum(abs(rstandard(fit)[influ_idx1]) > 2)
data2014_new <- data2014[-influ_idx1,]
interaction_model1 <- lm(expected_return~volatility, data=data2014_new)
plot(expected_return~volatility, data=data2014_new, pch=20,
      main = "2014 Annual Expected Return vs Volatility
            --Simple linear regression(Outliers removed)",cex.main=1)
abline(interaction_model1,col=2,lwd=2)
mean((data2014_new$expected_return - fitted(interaction_model1)) ^ 2)

# cubic root transformation
# installing the pracma package first
install.packages("pracma")
cubic_y <- pracma::nthroot(data2014_new$expected_return,9)
fit_cubic_y <- lm(cubic_y~data2014_new$volatility)
plot(cubic_y~data2014_new$volatility,pch=20,
      main = "2014 Cubic root Expected Return vs Volatility
            --(y^(1/3)~x)",cex.main=1)
abline(fit_cubic_y,col=2,lwd=2)
summary(fit_cubic_y)
mean((data2014_new$expected_return - fitted(fit_cubic_y)^3) ^ 2)
plot(expected_return~volatility,data=data2014_new,pch=20,
      main = "2014 Annual Expected Return vs Volatility
            --(y^(1/3)~x)",cex.main=1)
curve((fit_cubic_y$coef[1]+fit_cubic_y$coef[2]*x)^3,add=TRUE,col=2)

```

```

# split y into positive and negative
pos2014 <- data2014_new[data2014_new$expected_return > 0, ]
neg2014 <- data2014_new[data2014_new$expected_return < 0, ]
#order1
fit1_pos <- lm(expected_return~volatility,pos2014)
plot(expected_return~volatility,data=pos2014,pch=20)
abline(fit1_pos,col=2, lwd=2)
pos_mse <- mean((pos2014$expected_return - fitted(fit1_pos)) ^ 2)
fit1_neg <- lm(expected_return~volatility,neg2014)
plot(expected_return~volatility,data=neg2014,pch=20)
abline(fit1_neg,col=2, lwd=2)
neg_mse <- mean((neg2014$expected_return - fitted(fit1_neg)) ^ 2)
mse <- (131*pos_mse+102*neg_mse)/233
plot(expected_return~volatility,data=data2014_new,pch=20,
      main = "2014 Annual Expected Return vs Volatility (p=1)",cex.main=1)
curve(fit1_pos$coef[1]+fit1_pos$coef[2]*x,add=TRUE,col=2)
curve(fit1_neg$coef[1]+fit1_neg$coef[2]*x,add=TRUE,col=2)

#order3
y3_pos <- pracma::nthroot(pos2014$expected_return,3)
fit2_pos <- lm(y3_pos~pos2014$volatility)
plot(y3_pos~pos2014$volatility,pch=20)
abline(fit2_pos,col=2, lwd=2)
pos3_mse <- mean((pos2014$expected_return - fitted(fit2_pos)^3) ^ 2)
y3_neg <- pracma::nthroot(neg2014$expected_return,3)
fit2_neg <- lm(y3_neg~neg2014$volatility)
plot(y3_neg~neg2014$volatility,pch=20)
abline(fit2_neg,col=2, lwd=2)
neg3_mse <- mean((neg2014$expected_return - fitted(fit2_neg)^3) ^ 2)
mse3 <- (131*pos3_mse+102*neg3_mse)/233
plot(expected_return~volatility,data=data2014_new,pch=20,
      main = "2014 Annual Expected Return vs Volatility (p=3)",cex.main=1)
curve((fit2_pos$coef[1]+fit2_pos$coef[2]*x)^3,add=TRUE,col=2)
curve((fit2_neg$coef[1]+fit2_neg$coef[2]*x)^3,add=TRUE,col=2)

#order9
y9_pos <- pracma::nthroot(pos2014$expected_return,9)
fit3_pos <- lm(y9_pos~pos2014$volatility)
plot(y9_pos~pos2014$volatility,pch=20)
abline(fit3_pos,col=2, lwd=2)
pos9_mse <- mean((pos2014$expected_return - fitted(fit3_pos)^9) ^ 2)
y9_neg <- pracma::nthroot(neg2014$expected_return,9)
fit3_neg <- lm(y9_neg~neg2014$volatility)
plot(y9_neg~neg2014$volatility,pch=20)
abline(fit3_neg,col=2, lwd=2)
neg9_mse <- mean((neg2014$expected_return - fitted(fit3_neg)^9) ^ 2)
mse9 <- (131*pos9_mse+102*neg9_mse)/233
plot(expected_return~volatility,data=data2014_new,pch=20,
      main = "2014 Annual Expected Return vs Volatility (p=9)",cex.main=1)
curve((fit3_pos$coef[1]+fit3_pos$coef[2]*x)^9,add=TRUE,col=2)
curve((fit3_neg$coef[1]+fit3_neg$coef[2]*x)^9,add=TRUE,col=2)

```

```

#polynomial
fit_poly2_pos <- lm(expected_return~volatility+I(volatility^2), data=pos2014)
fit_poly2_neg <- lm(expected_return~volatility+I(volatility^2), data=neg2014)
poly2_mse <- (sum((pos2014$expected_return - fitted(fit_poly2_pos)) ^ 2)+
              sum((neg2014$expected_return - fitted(fit_poly2_neg)) ^ 2))/233
fit_poly3_pos <- lm(expected_return~volatility+I(volatility^2)+I(volatility^3), data=pos2014)
fit_poly3_neg <- lm(expected_return~volatility+I(volatility^2)+I(volatility^3), data=neg2014)
poly3_mse <- (sum((pos2014$expected_return - fitted(fit_poly3_pos)) ^ 2)+
              sum((neg2014$expected_return - fitted(fit_poly3_neg)) ^ 2))/233

plot(expected_return~volatility, data=data2014_new, pch=20,
      main = "2014 Annual Expected Return vs Volatility
      -- Polynomial Regression based on y", cex.main = 1)
abline(fit1_pos,lty = 2, col = "green", lwd = 2)
abline(fit1_neg,lty = 2, col = "green", lwd = 2)
xplot <- seq(0,0.8,by = 0.001)
lines(xplot, predict(fit_poly2_pos, newdata = data.frame(volatility = xplot)),
      col = "blue", lwd = 2)
lines(xplot, predict(fit_poly2_neg, newdata = data.frame(volatility = xplot)),
      col = "blue", lwd = 2)
lines(xplot, predict(fit_poly3_pos, newdata = data.frame(volatility = xplot)),
      col = "red", lty = 3, lwd = 2)
lines(xplot, predict(fit_poly3_neg, newdata = data.frame(volatility = xplot)),
      col = "red", lty = 3, lwd = 2)
legend("topright", legend = c("Poly1", "Poly2", "Poly3"),
      lwd = 2, lty = c(2, 1, 3), col = c("green", "blue", "red"),cex=0.8)

#log
fit_log_pos <- lm(expected_return~log(volatility),pos2014)
summary(fit_log_pos)
plot(expected_return~log(volatility),data=pos2014,pch=20)
abline(fit_log_pos,col=2, lwd=2)
fit_log_neg <- lm(expected_return~log(volatility),neg2014)
summary(fit_log_neg)
plot(expected_return~log(volatility),data=neg2014,pch=20)
abline(fit_log_neg,col=2, lwd=2)
log_mse <- (sum((pos2014$expected_return - fitted(fit_log_pos)) ^ 2)+
            sum((neg2014$expected_return - fitted(fit_log_neg)) ^ 2))/233
plot(expected_return~volatility,data=data2014_new,pch=20,
      main = "2014 Annual Expected Return vs Volatility
      -- y~log(x) based on y",cex.main=1)
curve(fit_log_pos$coef[1]+fit_log_pos$coef[2]*log(x),add=TRUE,col=2)
curve(fit_log_neg$coef[1]+fit_log_neg$coef[2]*log(x),add=TRUE,col=2)

```