```r
library(readxl) # Loading the `readxl` library to read Excel files
da=read_excel("/content/DATASET_HUGO.xlsx") # Reading the Excel file named "DATASET_HUGO.xlsx" and storing the data in the object `da`
da=data.frame(da) # Converting the object `da` into a data frame to ensure compatibility with data manipulation functions
head(da) # Displaying the first rows of the data frame for a quick overview of the data
```

```
  DATE       SANDP  IBM   CPI   INDPRO  M1SUPPLY CCREDIT  BMINUSA
USTB3M
1 1986-03-01 238.90 37.88 108.8 56.5414 624.3    606.7990 1.50    6.76

2 1986-04-01 235.52 39.06 108.6 56.5654 647.0    614.3669 1.40    6.24

3 1986-05-01 247.35 38.09 108.9 56.6850 645.7    621.9152 1.20    6.33

4 1986-06-01 250.84 36.62 109.5 56.4959 662.8    627.8910 1.21    6.40

5 1986-07-01 236.12 33.12 109.5 56.8096 673.4    633.6083 1.28    6.00

6 1986-08-01 252.93 34.69 109.7 56.7348 678.4    640.5126 1.46    5.69

  USTB10Y Adobe
1 7.78    0.09
2 7.30    0.11
3 7.71    0.13
4 7.80    0.12
5 7.30    0.14
6 7.17    0.16
```

```r
dim(da) # Returns the dimensions of the data frame `da`: the number of rows (observations) and the number of columns (variables)
```

```
[1] 385  11
```

```r
summary(da) # Generates a statistical summary of the columns in the data frame `da`, including statistics like mean, median, minimum and maximum values, and quartiles for numerical data. For categorical columns, it shows the frequency of each category
```

```
      DATE                         SANDP             IBM
 Min.   :1986-03-01 00:00:00.0   Min.   : 230.3   Min.   : 10.50
 1st Qu.:1994-03-01 00:00:00.0   1st Qu.: 459.3   1st Qu.: 30.00
 Median :2002-03-01 00:00:00.0   Median :1104.5   Median : 87.07
 Mean   :2002-03-02 01:11:03.8   Mean   :1066.0   Mean   : 87.34
 3rd Qu.:2010-03-01 00:00:00.0   3rd Qu.:1385.6   3rd Qu.:127.98
 Max.   :2018-03-01 00:00:00.0   Max.   :2823.8   Max.   :208.65
      CPI            INDPRO          M1SUPPLY        CCREDIT
 Min.   :108.6   Min.   : 56.50   Min.   : 624.3   Min.   : 606.8
 1st Qu.:147.2   1st Qu.: 69.48   1st Qu.:1069.3   1st Qu.: 886.2
 Median :178.8   Median : 93.00   Median :1191.8   Median :1891.8
```

```
 Mean   :181.1   Mean   : 86.63   Mean   :1514.7   Mean   :1897.8
 3rd Qu.:218.2   3rd Qu.:100.72   3rd Qu.:1716.0   3rd Qu.:2620.5
 Max.   :249.6   Max.   :106.66   Max.   :3684.7   Max.   :3843.4
    BMINUSA           USTB3M          USTB10Y          Adobe
 Min.   :0.5500   Min.   :0.010   Min.   :1.500   Min.   :  0.09
 1st Qu.:0.7200   1st Qu.:0.450   1st Qu.:3.330   1st Qu.:  3.66
 Median :0.9000   Median :3.440   Median :4.910   Median : 18.62
 Mean   :0.9746   Mean   :3.297   Mean   :5.075   Mean   : 28.50
 3rd Qu.:1.1300   3rd Qu.:5.290   3rd Qu.:6.740   3rd Qu.: 34.95
 Max.   :3.3800   Max.   :9.140   Max.   :9.520   Max.   :216.08
```

```
sandp=da[,2] # Extracting the 2nd column from the data frame `da`,
assumed to represent S&P data, and storing it in the object `sandp`
IBM=da[,3] # Extracting the 3rd column from `da`, assumed to contain
data for IBM, and storing it in the object `IBM`
cpi=da[,4] # Extracting the 4th column from `da`, assumed to represent
the Consumer Price Index (CPI), and storing it in the object `cpi`
indpro=da[,5] # Extracting the 5th column from `da`, assumed to
contain the industrial production index, and storing it in the object
`indpro`
m1supply=da[,6] # Extracting the 6th column from `da`, assumed to
represent M1 money supply, and storing it in the object `m1supply`
ccredit=da[,7] # Extracting the 7th column from `da`, assumed to
contain consumer credit data, and storing it in the object `ccredit`
bminusa=da[,8] # Extracting the 8th column from `da`, assumed to
represent the US balance of trade, and storing it in the object
`bminusa`
ustb3bm=da[,9] # Extracting the 9th column from `da`, assumed to
represent the 3-month US Treasury bill yield, and storing it in the
object `ustb3bm`
ustb10y=da[,10] # Extracting the 10th column from `da`, assumed to
contain the 10-year US Treasury bond yield, and storing it in the
object `ustb10y`
adobe=da[,11] # Extracting the 11th column from `da`, assumed to
contain data on Adobe, and storing it in the object `adobe`
```

This section of the code calculates the logarithmic return series and variations of several key economic variables to understand their percentage changes over time. Below is a detailed breakdown of each variable and its purpose:

- rsandp: The percentage change in the logarithmic return of the S&P 500 index, representing overall market performance.
- IBM: The percentage change in the logarithmic return of IBM stock, used to analyze the performance of IBM relative to the market.
- inflation: The inflation rate is calculated as the logarithmic change in the Consumer Price Index (CPI), providing insights into inflationary pressures in the economy.

- dspread: The difference or variation in the spread, which could refer to the difference between bond yields or other financial metrics, depending on the specific redefinition used in the model.
- dcredit: The change in credit conditions, typically represented by the variable 'ccredit', reflecting shifts in lending or borrowing activity.
- dmoney: The change in the money supply, typically measured by the M1 supply, indicating liquidity and monetary policy influences on the economy.
- dinflation: The variation in the inflation rate, tracking changes in the pace of inflation over time.
- term: The evolution of the yield curve, represented by the difference between the yields of 10-year bonds and 3-month Treasury bills, which gives insight into expectations for economic growth and interest rates.
- adobe: The percentage change in the logarithmic return of Adobe stock, providing an additional market-specific measure for analysis alongside IBM and S&P returns.

```
rsandp=c(NA,100*diff(log(sandp))) # Calculates the logarithmic return
series for S&P 500, converted into percentage changes
rIBM=c(NA,100*diff(log(IBM))) # Computes the logarithmic return series
for IBM, representing percentage growth rates
inflation=c(NA,100*diff(log(cpi))) # Computes the inflation rate as
the percentage change in the CPI (Consumer Price Index)
dspread=c(NA,100*diff(bminusa)) # Calculates the percentage change in
the trade balance spread over time
dcredit=c(NA,100*diff(ccredit)) # Computes the percentage change in
consumer credit data for each time period
dspread=c(NA,100*diff(indpro)) # Computes the percentage change in the
industrial production index over time
dmoney=c(NA,100*diff(m1supply)) # Calculates the percentage change in
the M1 money supply across time periods
dinflation=c(NA,100*diff(inflation)) # Computes the percentage change
in the inflation rate across consecutive periods
rterm=c(NA,100*diff(ustb10y-ustb3bm)) # Computes the percentage change
in the term spread (10-year vs 3-month Treasury yields)
radobe=c(NA,100*diff(log(adobe))) # Calculates the logarithmic return
series for Adobe, expressed as percentage changes
```

The following steps involve adjusting and transforming the interest rate data to align with the frequency of the stock returns and then calculating excess returns for both the market (S&P 500) and IBM.

- ustb3mmonth: This line divides the annual yield on 3-month Treasury bills (ustb3m) by 12 to convert it into a monthly yield. This adjustment allows for a more accurate comparison between the monthly-frequency returns of stocks and the monthly risk-free rate.

- ersandp: The excess return of the S&P 500 is calculated by subtracting the monthly risk-free rate (ustb3mmonth) from the S&P 500 return (rsandp). This helps in understanding the performance of the market above the risk-free rate, offering a measure of the market's risk-adjusted return.
- erIBM: Similarly, the excess return of IBM is computed by subtracting the monthly risk-free return from IBM's stock return. This allows for the analysis of IBM's performance relative to a risk-free investment, adjusting for the broader market's risk-free rate.

This entire process is aimed at providing a detailed analysis of the relationship between various economic indicators and the performance of financial assets.

The code calculates the monthly yield for 3-month Treasury bills (`ustb3bmmonth`) by dividing the annual yield (`ustb3bm`) by 12. It then computes the excess return for the S&P 500 (`ersandp`) by subtracting the monthly risk-free rate (`ustb3bmmonth`) from the S&P 500 return (`rsandp`). Similarly, the excess return for IBM (`erIBM`) is calculated by subtracting the same monthly risk-free rate (`ustb3bmmonth`) from IBM's return (`rIBM`). This allows for an analysis of the stock returns relative to the risk-free rate over the same period.

```
ustb3bmmonth=ustb3bm/12 # Calculates the monthly yield for 3-month Treasury bills by dividing the annual yield by 12
ersandp=rsandp-ustb3bmmonth # Calculates the excess return of S&P 500 by subtracting the risk-free rate
erIBM=rIBM-ustb3bmmonth # Calculates the excess return of IBM by subtracting the risk-free rate
```

This R code analyzes economic and financial data from the Excel file **"DATASET_HUGO.xlsx"**, calculating logarithmic returns, percentage changes, inflation, interest rate spreads, and the **excess returns** of S&P 500 and IBM stocks. The goal is to explore the relationships between these indicators and financial asset performance.

```
install.packages('lmtest') # Installs the 'lmtest' package, which provides tools for diagnostic checking in linear models
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
also installing the dependency ‘zoo’
```

```
library(lmtest) # Loads the 'lmtest' package to use functions for linear model diagnostics and hypothesis testing
```

```
Loading required package: zoo
```

```
Attaching package: ‘zoo’
```

```
The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric
```

The following code specifies a multiple linear model (with the lm function) to explain IBM's excess return (rIBM) as a function of several explanatory variables.

The model seeks to estimate the contributions of each explanatory variable to IBM's excess return (rIBM).

```
modelapt=lm(erIBM ~ ersandp + dcredit + dinflation + dmoney + dspread
+ rterm, data=da) # Creates a linear regression model with 'rIBM' as
the dependent variable and the other variables as independent
variables
summary(modelapt) # Displays a summary of the regression model,
including coefficients, standard errors, and p-values


Call:
lm(formula = erIBM ~ ersandp + dcredit + dinflation + dmoney +
    dspread + rterm, data = da)

Residuals:
     Min       1Q   Median       3Q      Max
-30.2899  -3.3548  -0.3086   3.7686  21.9081

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.928e-02  3.795e-01  -0.051   0.9595
ersandp      9.610e-01  7.395e-02  12.997   <2e-16 ***
dcredit     -4.338e-05  2.166e-04  -0.200   0.8414
dinflation   7.975e-03  9.834e-03   0.811   0.4179
dmoney      -1.126e-04  1.230e-04  -0.916   0.3604
dspread     -9.821e-03  5.864e-03  -1.675   0.0948 .
rterm        2.629e-02  1.371e-02   1.918   0.0559 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.268 on 376 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.3192,    Adjusted R-squared:  0.3083
F-statistic: 29.38 on 6 and 376 DF,  p-value: < 2.2e-16
```

The coeftest(modelapt) function is part of the lmtest package and is used to perform hypothesis tests on the coefficients of the regression model modelapt. It provides the following key statistics:

- Coefficient Estimates (β): These are the estimated values for each independent variable, indicating their effect on the dependent variable.
- Standard Errors: These measure the precision of the coefficient estimates, showing how much the estimates might vary across different samples.
- t-Values: These are test statistics used to assess the significance of each coefficient by comparing the estimate to its standard error.
- p-Values: These values help determine whether each coefficient is statistically significant, with smaller p-values (typically < 0.05) suggesting that the null hypothesis (coefficient = 0) can be rejected.

```
coeftest(modelapt) # Displays the results of significance tests for
the coefficients of the 'modelapt', including t-values and p-values


t test of coefficients:

               Estimate  Std. Error t value Pr(>|t|)
(Intercept) -1.9278e-02  3.7948e-01 -0.0508  0.95951
ersandp      9.6103e-01  7.3945e-02 12.9965  < 2e-16 ***
dcredit     -4.3381e-05  2.1659e-04 -0.2003  0.84136
dinflation   7.9752e-03  9.8341e-03  0.8110  0.41789
dmoney      -1.1264e-04  1.2300e-04 -0.9158  0.36038
dspread     -9.8206e-03  5.8637e-03 -1.6748  0.09480 .
rterm        2.6289e-02  1.3706e-02  1.9181  0.05586 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

install.packages('car') # Installs the 'car' package, which provides
functions for regression analysis and diagnostic tools for linear
models

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘cowplot’, ‘Deriv’, ‘microbenchmark’,
‘numDeriv’, ‘doBy’, ‘SparseM’, ‘MatrixModels’, ‘minqa’, ‘nloptr’,
‘RcppEigen’, ‘carData’, ‘abind’, ‘Formula’, ‘pbkrtest’, ‘quantreg’,
‘lme4’
```

The following code utilizes the linearHypothesis function from the car package to conduct a simultaneous hypothesis test on the coefficients of the modelapt regression model. Here's a breakdown of the procedure:

Objective: The goal is to test whether the specified coefficients in the regression model are simultaneously equal to zero.

Hypothesis:

- Null hypothesis (H_0): The coefficients for dcredit, dmoney, and dspread are all equal to 0.
- Alternative hypothesis (H_1): At least one of these coefficients is different from 0.

F-test Statistic: This statistic evaluates whether the combined effect of the coefficients being tested significantly contributes to the model's explanatory power.

Interpretation of p-value:

- Small p-value (< 0.05): Reject the null hypothesis (H_0), indicating that at least one of the variables has a significant effect on the model.
- Large p-value (≥ 0.05): Fail to reject the null hypothesis (H_0), suggesting that the combined effect of the variables is not statistically significant.

```
library(car) # Loads the 'car' package, which contains tools for
regression diagnostics and hypothesis testing for linear models
linearHypothesis(modelapt, c("dcredit=0", "dmoney=0","dspread=0")) #
Performs a linear hypothesis test on the 'modelapt' to check if the
coefficients of 'dcredit', 'dmoney', and 'dspread' are equal to zero
```

```
Loading required package: carData
```

```
  Res.Df RSS      Df Sum of Sq F      Pr(>F)
1 379    14915.37 NA       NA      NA        NA
2 376    14771.41  3 143.9623  1.2215 0.3016259
```

```
install.packages("quantreg") # Installs the 'quantreg' package, which
provides tools for quantile regression and robust quantile estimation
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

We import the library 'quantreg' to estimate the quantile

```
library(quantreg) # Loads the 'quantreg' package, which allows
performing quantile regressions and estimating robust models for
different quantiles
```

```
Loading required package: SparseM
```

Quantile Regression: Unlike traditional linear regression, which models the conditional mean of the dependent variable, quantile regression estimates conditional quantiles, such as the median or other percentiles.

This method is particularly valuable for understanding how explanatory variables impact different segments of the dependent variable's distribution, allowing for insights into the effects on the central tendency as well as on the tails (e.g., extreme values) of the distribution.

```
qregcapm=rq(erIBM ~ ersandp, data=da) # Performs a quantile regression (rq) to estimate the relationship between 'rIBM' and 'rsandp' using the data from the 'da' dataframe

summary(qregcapm) # Displays a summary of the quantile regression results 'qregcapm', including coefficients, standard errors, and test statistics

Warning message in rq.fit.br(x, y, tau = tau, ci = TRUE, ...):
“Solution may be nonunique”


Call: rq(formula = erIBM ~ ersandp, data = da)

tau: [1] 0.5

Coefficients:
            coefficients lower bd upper bd
(Intercept) -0.41497     -0.83263  0.07315
ersandp      0.96027      0.86309  1.08453
```

The following code carries out quantile regression across various quantiles of the conditional distribution. By estimating the regression at several quantiles (τ), it allows us to explore how the relationship between `rIBM` and `rsandp` differs across different parts of the distribution.

For instance:

τ=0.1: Focuses on analyzing IBM's low returns, conditional on the S&P 500.

τ=0.5: Examines the median, providing insight into the central tendency of the data.

τ=0.9: Investigates IBM's high returns, reflecting the upper tail of the distribution.

The coefficients at each quantile (τ) show how Apple's excess returns respond to variations in the S&P 500 under specific market conditions, such as bear, normal, or bull markets.

```
rq(erIBM ~ ersandp, data=da, tau=seq(0.1,0.9,0.1)) # Performs a quantile regression for different quantiles (tau from 0.1 to 0.9) of 'rIBM' based on 'rsandp', using the data from the 'da' dataframe

Call:
rq(formula = erIBM ~ ersandp, tau = seq(0.1, 0.9, 0.1), data = da)

Coefficients:
              tau= 0.1   tau= 0.2   tau= 0.3   tau= 0.4   tau= 0.5
tau= 0.6
(Intercept) -7.0819585 -4.4619301 -2.9043659 -1.7418680 -0.4149691
0.5137591
```

```
ersandp      0.9237182  0.9669927  0.9970094  0.9230995  0.9602653
0.9666551
             tau= 0.7  tau= 0.8  tau= 0.9
(Intercept) 2.2573343 4.2595713 7.7151286
ersandp     0.9167838 0.9536269 0.8682498

Degrees of freedom: 384 total; 382 residual
```

The following code performs quantile regression on IBM's excess returns (rIBM) as a function of the S&P 500's excess returns (rsandp) across various quantiles.

rq(): The quantile regression function from the quantreg package. It estimates relationships between variables at different quantiles of the dependent variable (here, rIBM).

rIBM ~ rsandp: This formula specifies that we are modeling rIBM as a function of rsandp. It allows us to analyze how the S&P 500's excess returns influence IBM's excess returns.

data=da: Specifies that the data used comes from the dataset 'da'.

tau=seq(0.1, 0.9, 0.1): Defines a range of quantiles from τ=0.1 to τ=0.9 in steps of 0.1. This ensures that the regression is estimated for the 10th, 20th, ..., 90th quantiles of IBM's excess returns, relative to the S&P 500.

The resulting `qreg` model will be a list of quantile regression models, each corresponding to a different quantile (τ) specified. This setup allows us to examine how the relationship between rsandp and rIBM varies across different parts of the distribution of IBM's excess returns.

Each regression model targets a specific quantile of IBM's excess returns distribution. For example:

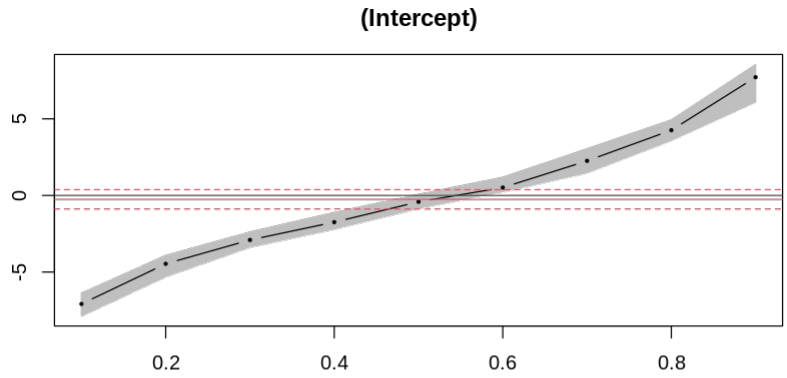
- **τ=0.1**: Analyzes the relationship between rsandp and rIBM during periods of low IBM returns.
- **τ=0.5**: Focuses on the median relationship, providing insight into the central tendency.
- **τ=0.9**: Investigates how rsandp affects IBM during high-return periods.

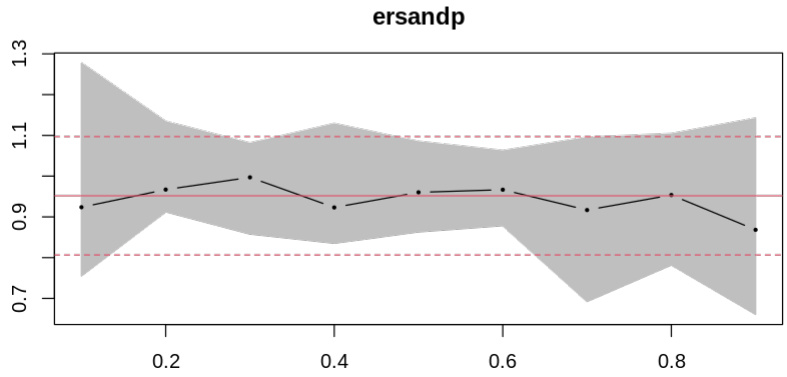
This method is especially useful when the effects of explanatory variables, like the S&P 500's excess returns, differ across the distribution of returns—such as stronger effects during bull or bear markets.

```
qreg= rq(erIBM ~ ersandp, data=da, tau=seq(0.1,0.9,0.1)) # Performs a
quantile regression for different quantiles (tau from 0.1 to 0.9) of
'rIBM' based on 'rsandp', and stores the result in the 'qreg' object

plot(summary(qreg), level = 0.95) # Plots a graph of the quantile
regression results 'qreg', displaying the 95% confidence intervals for
the estimated coefficients

Warning message in rq.fit.br(x, y, tau = tau, ci = TRUE, ...):
"Solution may be nonunique"
```





The intercept in the OLS linear regression model is fixed and centralized, typically around -0.008 in the context of the CAPM model. This fixed intercept does not fully capture the dynamic nature of returns, especially those observed in the extremes or tails of the distribution. By focusing solely on the average relationship, OLS misses the varying impacts that market conditions can have on the returns of an asset, particularly in volatile or extreme market environments.

In contrast, quantile regression allows for a more adaptive approach. It estimates different intercepts for various quantiles of the distribution (e.g., 10th, 50th, 90th percentiles), meaning it can capture how the relationship between the independent variable (market returns) and the dependent variable (IBM's excess returns) changes across different segments of the return distribution. This flexibility is particularly valuable for understanding extreme market behaviors, whether in bear or bull markets.

Conclusion: Quantile regression provides a more comprehensive analysis of the relationship between IBM's excess returns and the market returns, as it enables a deeper understanding of how this relationship differs under various market conditions. Unlike OLS, which offers a single average measure, quantile regression reveals how market movements impact the asset's returns at different points in its distribution, offering more insights into extreme market events and the behavior of returns in various economic contexts.

```
dev.print(device=pdf,file="rqcapm.pdf") # Saves the current plot in a
PDF file named 'rqcapm.pdf'

pdf
  2
```

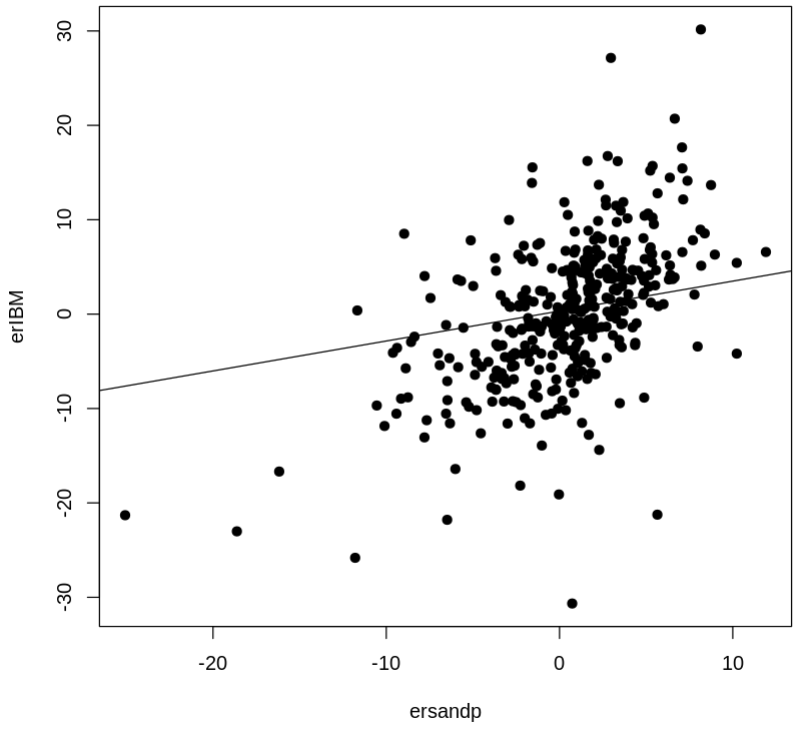
The following lines of code calculate the logarithmic returns for two stocks, Ford and Adobe, based on their price data. Here's a breakdown of what each line does:

- **log(IBM) and log(adobe)**: This function computes the natural logarithm of the IBM and Adobe stock prices. Using logarithms to calculate returns normalizes the price variations and allows them to be compared independently of the absolute price level.
- **diff()**: This function calculates the difference between consecutive elements in the data series. In this context, it computes the logarithmic differences between the current and previous stock prices, yielding the logarithmic changes in stock prices.
- **Multiplying by 100**: The logarithmic differences are multiplied by 100 to express the results as percentage returns. This makes the interpretation of the changes more intuitive, as it represents the percentage variation in stock prices rather than the raw logarithmic values.
- **c(NA, ...)**: An "NA" value is prepended to the beginning of the return series to align its length with that of the original price data. Since the `diff()` function reduces the series length by one (as there is no predecessor for the first value), the NA ensures that the return series matches the stock price data in length.

```
rIBM=c(NA,100*diff(log(IBM))) # Computes the logarithmic return series
for 'IBM', expressed as a percentage, with a 'NA' added at the
beginning to align the data lengths
radobe=c(NA,100*diff(log(adobe))) # Computes the logarithmic return
series for 'adobe', expressed as a percentage, with a 'NA' added at
the beginning to align the data lengths

plot(ersandp,erIBM,pch=19) # Plots a scatter plot between 'rsandp' and
'rIBM', with filled points (pch=19)
abline(lm(ersandp~erIBM)) # Adds a linear regression line to the plot,
representing the relationship between 'rsandp' and 'rIBM'
dev.print(device=pdf,file="assoc.pdf") # Saves the current plot in a
PDF file named 'assoc.pdf'

pdf
  2
```



This code generates a scatter plot showing the relationship between the excess returns of the S&P 500 index (**rsandp**) and the excess returns of IBM (**rIBM**).

- The **plot()** function creates a scatter plot with filled points (pch=19), visually representing the data points.
- The **abline()** function adds a linear regression line to the plot, which represents the estimated relationship between the two variables (in this case, **rsandp** and **rIBM**). The regression line shows the direction and strength of the correlation between these two excess return series.
- The **dev.print()** function saves the plot as a PDF file called **assoc.pdf**, allowing for later review or presentation.

Interpretation: By examining the scatter plot and the linear regression line, you can assess whether there is a linear relationship between the S&P 500 and IBM's excess returns. A positive slope suggests that, as the S&P 500 excess returns increase, IBM's excess returns tend to rise as well. Conversely, a negative slope would suggest an inverse relationship.

```
install.packages('stargazer') # Installs the 'stargazer' package,
which allows creating regression result tables and other statistical
analysis in a clean and readable format
install.packages('officer') # Installs the 'officer' package, which
allows manipulating Word and PowerPoint documents in R
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(stargazer) # Loads the 'stargazer' package, which allows
easily creating regression result tables, descriptive statistics, etc.
library(officer) # Loads the 'officer' package, which allows creating,
modifying, and analyzing Word and PowerPoint documents directly from R
models =list(modelapt) # Creates a list containing the 'modelapt',
allowing multiple models to be stored in a data structure
stargazer(models, title = "Regression Results", type = "text", out =
"capm.doc") # Generates a regression result table and exports it to a
Word file named 'capm.doc'
```

```
Please cite as:


 Hlavac, Marek (2022). stargazer: Well-Formatted Regression and
Summary Statistics Tables.

 R package version 5.2.3. https://CRAN.R-project.org/package=stargazer



Attaching package: ‘officer’


The following object is masked from ‘package:readxl’:

    read_xlsx



Regression Results
================================================
                        Dependent variable:
                    ----------------------------
                               erIBM
------------------------------------------------
ersandp                       0.961***
                               (0.074)

dcredit                       -0.00004
                              (0.0002)
```

```
dinflation                      0.008
                               (0.010)

dmoney                         -0.0001
                              (0.0001)

dspread                        -0.010*
                               (0.006)

rterm                          0.026*
                               (0.014)

Constant                       -0.019
                               (0.379)

------------------------------------------------
Observations                     383
R2                              0.319
Adjusted R2                     0.308
Residual Std. Error      6.268 (df = 376)
F Statistic           29.378*** (df = 6; 376)
================================================
Note:                *p<0.1; **p<0.05; ***p<0.01
```

```
bptest(formula(modelapt),data=da,studentize=F) # Performs the Breusch-Pagan test for heteroscedasticity in the 'modelapt', without standardizing the residuals
```

```
	Breusch-Pagan test

data:  formula(modelapt)
BP = 5.7187, df = 6, p-value = 0.4554
```

```
bptest(formula(modelapt),data=da,studentize=T) # Performs the Breusch-Pagan test for heteroscedasticity in the 'modelapt', with standardizing the residuals
```

```
	studentized Breusch-Pagan test

data:  formula(modelapt)
BP = 2.6079, df = 6, p-value = 0.8562
```

```
install.packages("sandwich") # Installs the 'sandwich' package, which provides robust standard error estimations for regression models
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(sandwich) # Loads the 'sandwich' package, which allows
calculating robust standard errors for regression models

coeftest(modelapt,vcov.=vcovHC(modelapt,type="HC1")) # Performs a test
of the coefficients of the 'modelapt' using robust standard errors of
type "HC1" to correct for heteroscedasticity
```

```
t test of coefficients:

               Estimate  Std. Error t value Pr(>|t|)
(Intercept) -1.9278e-02  3.6367e-01 -0.0530  0.95775
ersandp      9.6103e-01  7.7138e-02 12.4586  < 2e-16 ***
dcredit     -4.3381e-05  1.5078e-04 -0.2877  0.77372
dinflation   7.9752e-03  1.0283e-02  0.7756  0.43847
dmoney      -1.1264e-04  9.1300e-05 -1.2337  0.21809
dspread     -9.8206e-03  5.7027e-03 -1.7221  0.08588 .
rterm        2.6289e-02  1.4288e-02  1.8399  0.06658 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
coeftest(modelapt,vcov.=NeweyWest(modelapt,lag = 6, adjust=T,
prewhite=F)) # Performs a test of the coefficients of the 'modelapt'
using Newey-West robust standard errors, with a lag of 6, adjusted for
variance and without pre-whitening
```

```
t test of coefficients:

               Estimate  Std. Error t value Pr(>|t|)
(Intercept) -1.9278e-02  3.4293e-01 -0.0562  0.95520
ersandp      9.6103e-01  9.1056e-02 10.5542  < 2e-16 ***
dcredit     -4.3381e-05  1.4501e-04 -0.2991  0.76499
dinflation   7.9752e-03  1.1306e-02  0.7054  0.48099
dmoney      -1.1264e-04  9.7518e-05 -1.1550  0.24881
dspread     -9.8206e-03  4.8156e-03 -2.0393  0.04212 *
rterm        2.6289e-02  1.4458e-02  1.8182  0.06983 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Alternative hypothesis: At least one of the coefficients is significantly different from zero, indicating a meaningful relationship between the predictor variables and the response variable.

Conclusion: The `coeftest` output reveals that the variable **rsandp** is highly significant (**p-value < 2e-16**), suggesting a strong relationship with the dependent variable. On the other hand, variables like **dcredit**, **dinflation**, and **dmoney** show no significant relationship with the dependent variable, as their p-values are much higher than 0.05. Additionally, **dspread** is statistically significant at the 5% level, while **rterm** is marginally significant (p-value = 0.07013). The use of Newey-West robust standard errors helps adjust for heteroscedasticity and autocorrelation, providing more reliable coefficient estimates.

```
dwtest(modelapt) # Performs the Durbin-Watson test to detect autocorrelation in the residuals of the 'modelapt'
```

```
	Durbin-Watson test

data:  modelapt
DW = 1.9934, p-value = 0.4623
alternative hypothesis: true autocorrelation is greater than 0
```

```
bgtest(modelapt,order=10) # Performs the Breusch-Godfrey test to detect autocorrelation in the residuals of the 'modelapt', using an order of 10 lags
```

```
	Breusch-Godfrey test for serial correlation of order up to 10

data:  modelapt
LM test = 12.741, df = 10, p-value = 0.2385
```

```
install.packages("moments") # Installs the 'moments' package, which provides functions to compute statistical moments like skewness and kurtosis
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(moments) # Loads the 'moments' package, which allows computing distribution measures such as skewness and kurtosis
```

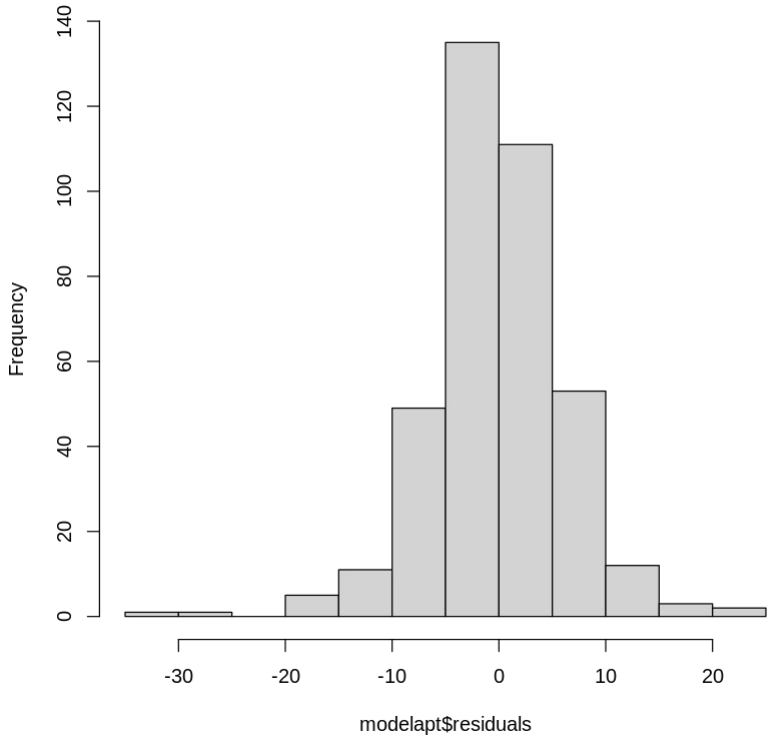
```
skewness(modelapt$residuals) # Computes the skewness of the residuals of the 'modelapt' to assess the symmetry of the error distribution
```

```
[1] -0.2250687
```

```
kurtosis(modelapt$residuals) # Computes the kurtosis of the residuals of the 'modelapt' to assess the "pointedness" or flatness of the error distribution
```

```
[1] 5.385651
```

```
hist(modelapt$residuals,main="") # Plots a histogram of the residuals of the 'modelapt', with no specified title (main="")
```




```
jarque.test(modelapt$residuals) # Performs the Jarque-Bera test to
test the normality of the residuals of the 'modelapt', assessing
skewness and kurtosis
```

```
     Jarque-Bera Normality Test

data:  modelapt$residuals
JB = 94.058, p-value < 2.2e-16
alternative hypothesis: greater
```

Alternative hypothesis: The residuals do not follow a normal distribution, which suggests that the data may exhibit skewness or kurtosis deviations from the normal distribution.

Conclusion: The test rejects the null hypothesis of normality (p-value < 2.2e-16), indicating that the residuals of the `modelapt` do not follow a normal distribution. This suggests potential issues with model specification or that the data contains non-normal features, such as skewness or excess kurtosis.

```
agostino.test(modelapt$residuals) # Performs the Agostino test (or
D'Agostino test) to test the normality of the residuals of the
'modelapt', based on the moments of the distribution
```

```
	D'Agostino skewness test

data:  modelapt$residuals
skew = -0.22507, z = -1.81065, p-value = 0.0702
alternative hypothesis: data have a skewness
```

```
anscombe.test(modelapt$residuals) # Performs the Anscombe test to assess the normality of the residuals of the 'modelapt', using a method based on symmetry and the distribution of the errors
```

```
	Anscombe-Glynn kurtosis test

data:  modelapt$residuals
kurt = 5.3857, z = 5.0593, p-value = 4.208e-07
alternative hypothesis: kurtosis is not equal to 3
```

Alternative hypothesis: The kurtosis is not equal to 3, suggesting that the residuals do not follow a normal distribution. This could indicate heavier tails (the model exhibits extreme values more frequently than a normal model).

Conclusion: The test rejects the null hypothesis (kurtosis = 3), implying that the residuals of the `modelapt` do not follow a normal distribution. This may signal non-normality of the errors and potentially indicate a modeling issue or variables that are not captured in the model.

```

```