Phụ lục

Phần phụ lục này tập hợp các bài viết độc lập hoặc nối tiếp các ngày trong nội dung chính của eBook. Đây là phần chia sẻ thêm để giúp bạn đọc có nhu cầu thực hành, khám phá thêm các góc nhìn mới về Phân tích dữ liệu nói riêng, AI nói chung.

Các bài viết có thể ở dạng ý tưởng và đang trong giai đoạn khám phá. Vì vậy nếu bạn đọc thấy chưa hoàn thiện thì đó xem như bài tập của các bạn nhé!

Thân ái,

Chỉ số VN INDEX biến động như thế nào từ thứ Hai đến thứ Sáu

Tôi có câu hỏi là nên giao dịch chứng khoáng vào thứ mấy trong tuần?

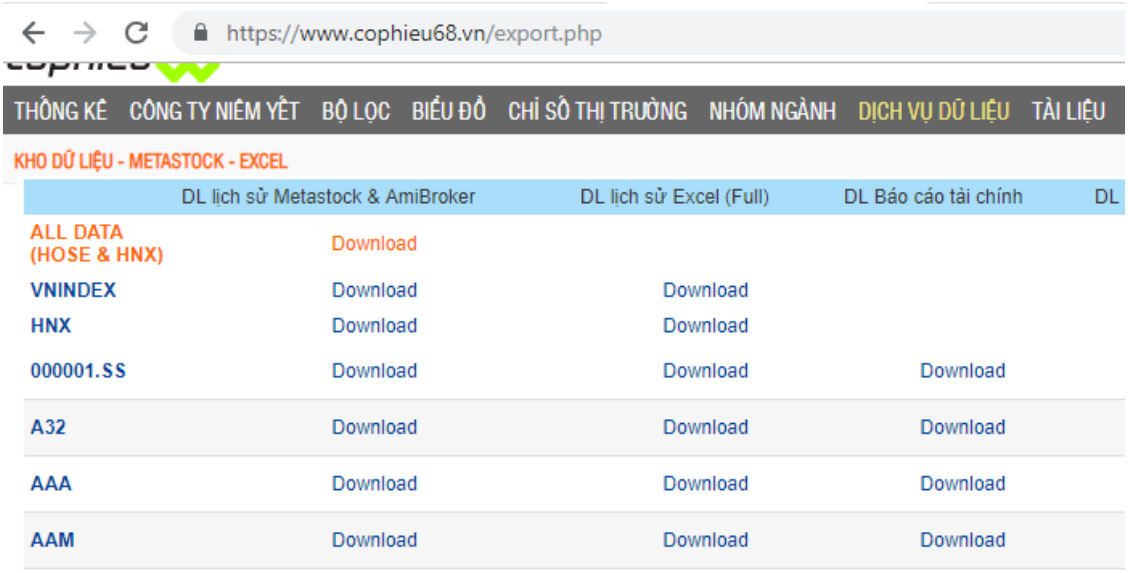
Để trả lời câu hỏi này thì chúng ta cần giải quyết một số vấn đề sau:

- Xem dữ liệu VNIndex giá trị High, Low các ngày trong tuần

Bài viết này minh họa bằng code R.

Cách lấy dữ liệu VN INDEX

Vào trang web "https://www.cophieu68.vn/export.php", đăng ký tài khoản (miễn phí) bấm vào mục "DỊCH VỤ DỮ LIỆU", "Tải dữ liệu" để lấy file csv về máy.



Dùng phần mềm R (https://cran.r-project.org/) để xem tổng quan dữ liệu.

Các dòng bắt đầu bằng dấu thăng (#) là chú thích của tôi để giải thích ý nghĩa của lệnh R.

```
# Dùng lệnh file.choose() để mở hộp thoại
# chọn file "excel_^vnindex.csv" sau khi download
f = file.choose()
data = read.csv(f)
dim(data)
```

```
[1] 4506   14
```

 # Kết quả lệnh dim cho thấy có 4506 dòng dữ liệu

 # 14 cột dữ liệu (Xem thêm lệnh head bên dưới)

```
head(data)
```

```
  X.Ticker. X.DTYYYYMMDD. X.OpenFixed. X.HighFixed. X.LowFixed. X.CloseFixed.
1  ^VNINDEX       20190426        972.86        979.64       970.73         979.64
2  ^VNINDEX       20190425        976.46        976.46       971.92         974.13
3  ^VNINDEX       20190424        969.66        978.71       969.66         976.92
4  ^VNINDEX       20190423        964.84        970.98       964.35         968.00
```

```
5  ^VNINDEX      20190422       963.78       966.69      959.33       965.86
6  ^VNINDEX      20190419       968.27       971.73      965.46       966.21
  X.Volume. X.Open. X.High. X.Low. X.Close. X.VolumeDeal. X.VolumeFB. X.VolumeFS.
1 118539754  972.86  979.64 970.73   979.64             0     6527232     6355132
2 149114784  976.46  976.46 971.92   974.13             0    15370270    16830400
3 134071764  969.66  978.71 969.66   976.92             0    12740050     7683390
4 154887616  964.84  970.98 964.35   968.00             0    11690700    18355300
5 216344286  963.78  966.69 959.33   965.86             0     6673120     6090200
6 106976093  968.27  971.73 965.46   966.21             0     3204780     4536850
```

Chú ý: Cột ngày "`X.DTYYYYMMDD.`" được trình bày theo dạng yyyyMMdd – không có dấu cách giữa năm, tháng ngày nên R hiểu đây là số nguyên. Dùng lệnh class để xem kiểu dữ liệu:

```
class(data$X.DTYYYYMMDD.)
```

```
[1] "integer"
```

Chúng ta cần chuyển đổi dữ liệu thời gian này một chút thông qua 2 bước:

Bước 1: Thêm cột strDate bằng cách lấy dữ liệu cột "`X.DTYYYYMMDD.`" chuyển thành kiểu kí tự (chuỗi).

```
data$strDate = as.character(data$X.DTYYYYMMDD.)
class(data$strDate)
```

```
[1] "character"
```

Bước 2: Thêm cột data bằng cấy lấy dữ liệu cột "strDate" vừa thêm chuyển thành kiểu ngày bằng hàm as.Date(strDate, '%Y%m%d)

```
data$date = as.Date(data$strDate, format = '%Y%m%d')
class(data$date)
```

```
[1] "Date"
```

Chú ý: trong định dạng %Y%d%m thì chữ **d và m là chữ thường**.

Kiểm tra lại vài dòng dữ liệu

```
head(data)
```

```
  X.Ticker. X.DTYYYYMMDD. X.OpenFixed. X.HighFixed. X.LowFixed. X.CloseFixed. X.Volume.
X.Open. X.High. X.Low. X.Close. X.VolumeDeal.
1  ^VNINDEX      20190426       972.86       979.64      970.73       979.64 118539754
972.86  979.64 970.73   979.64             0
2  ^VNINDEX      20190425       976.46       976.46      971.92       974.13 149114784
976.46  976.46 971.92   974.13             0
3  ^VNINDEX      20190424       969.66       978.71      969.66       976.92 134071764
969.66  978.71 969.66   976.92             0
4  ^VNINDEX      20190423       964.84       970.98      964.35       968.00 154887616
964.84  970.98 964.35   968.00             0
5  ^VNINDEX      20190422       963.78       966.69      959.33       965.86 216344286
963.78  966.69 959.33   965.86             0
6  ^VNINDEX      20190419       968.27       971.73      965.46       966.21 106976093
968.27  971.73 965.46   966.21             0
  X.VolumeFB. X.VolumeFS.  strDate       date
1     6527232     6355132 20190426 2019-04-26
2    15370270    16830400 20190425 2019-04-25
3    12740050     7683390 20190424 2019-04-24
4    11690700    18355300 20190423 2019-04-23
```

```
5     6673120     6090200 20190422 2019-04-22
6     3204780     4536850 20190419 2019-04-19
```

Lúc này dữ liệu cột date được hiển thị có dấu gạch giữa năm tháng và ngày.

Xem tổng quan dữ liệu bằng lệnh summary:

```
summary(data)
```

```
    X.Ticker.     X.DTYYYYMMDD.        X.OpenFixed.      X.HighFixed.      X.LowFixed.
X.CloseFixed.
 ^VNINDEX:4506   Min.   :20000728   Min.   : 100.0   Min.   : 100.0   Min.   : 100.0   M
in.   : 100.0
                 1st Qu.:20050926   1st Qu.: 296.5   1st Qu.: 298.9   1st Qu.: 296.3   1
st Qu.: 297.4
                 Median :20100412   Median : 487.6   Median : 490.6   Median : 484.0   M
edian : 486.9
                 Mean   :20098088   Mean   : 507.6   Mean   : 510.1   Mean   : 505.0   M
ean   : 507.5
                 3rd Qu.:20141019   3rd Qu.: 610.7   3rd Qu.: 614.4   3rd Qu.: 607.8   3
rd Qu.: 610.6
                 Max.   :20190426   Max.   :1207.6   Max.   :1211.3   Max.   :1197.4   M
ax.   :1204.3

   X.Volume.            X.Open.          X.High.          X.Low.          X.Close.
X.VolumeDeal.
 Min.   :      174   Min.   : 100.0   Min.   : 100.0   Min.   : 100.0   Min.   : 100.0
Min.   :0
 1st Qu.:  1487382   1st Qu.: 296.5   1st Qu.: 298.9   1st Qu.: 296.3   1st Qu.: 297.3
1st Qu.:0
 Median : 26702355   Median : 487.6   Median : 490.6   Median : 484.0   Median : 486.8
Median :0
 Mean   : 53381363   Mean   : 507.6   Mean   : 510.1   Mean   : 505.0   Mean   : 507.5
Mean   :0
 3rd Qu.: 93457508   3rd Qu.: 610.7   3rd Qu.: 614.4   3rd Qu.: 607.8   3rd Qu.: 610.7
3rd Qu.:0
 Max.   :445940510   Max.   :1207.6   Max.   :1211.3   Max.   :1197.4   Max.   :1204.3
Max.   :0
                                                                         NA's   :1

  X.VolumeFB.          X.VolumeFS.             date
 Min.   :0.000e+00   Min.   :0.000e+00   Min.   :2000-07-28
 1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:2005-09-26
 Median :2.540e+06   Median :2.140e+06   Median :2010-04-12
 Mean   :1.614e+07   Mean   :1.625e+07   Mean   :2010-04-02
 3rd Qu.:6.608e+06   3rd Qu.:5.979e+06   3rd Qu.:2014-10-19
 Max.   :2.147e+09   Max.   :1.847e+09   Max.   :2019-04-26
```

Ghi chú:

- Trong cột ‘date” cho biết dữ liệu từ ngày 28/7/2000 (dòng **Min.**) đến 26/4/2019 (Dòng **Max.**)

Thêm cột “dayOfWeek” để thể hiện Thứ trong tuần.

```
data$dayOfWeek = weekdays(date)
```

Chuyển dayOfWeek thành Factor để phục vụ cho việc phân tích

```
data$dayOfWeek = as.factor(data$dayOfWeek)
```

Xem lại tổng quan dữ liệu của dayOfWeek bằng lệnh summary bạn sẽ thấy số lượng dữ liệu theo thứ.

```
summary(data$dayOfWeek)
```

```
   Friday    Monday  Thursday   Tuesday Wednesday
      938       911       862       854       941
```

Vẽ biểu đồ với thư viện zoo

Zoo hỗ trợ phân tích dữ liệu theo thời gian.

Cài đặt thư viện zoo:

```
install.packages('zoo')
install.packages('ggfortify')
library(zoo)
library(ggfortify)
```

Tạo dữ liệu x theo thời gian của giá trị thấp nhất và cao nhất của VNIndex

```
z = zoo(x = cbind(X.Low., X.High.), order.by = date)
```

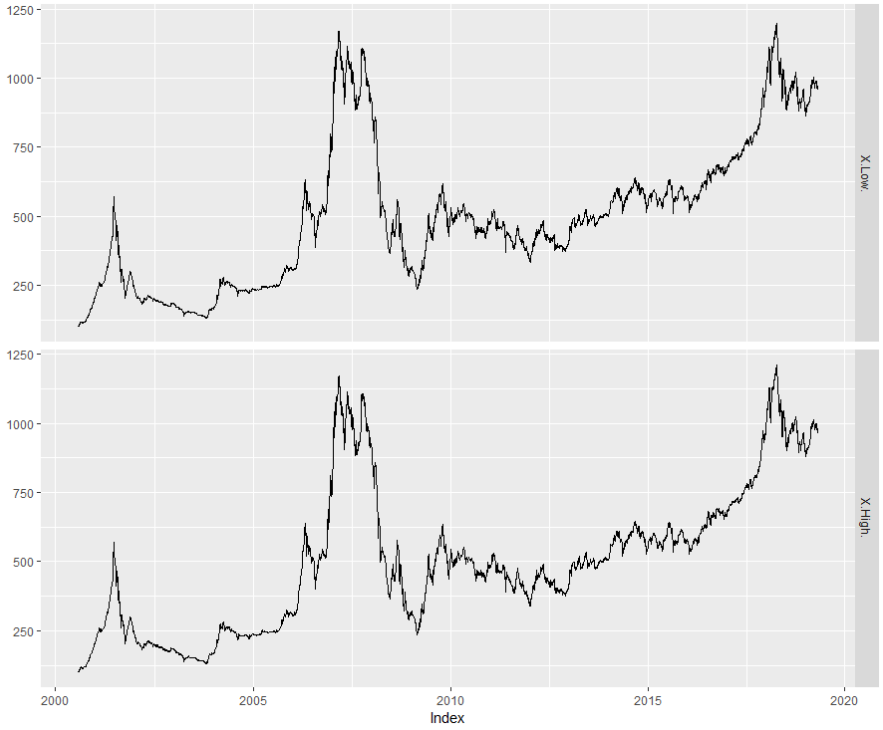
Nhìn qua dữ liệu của biến z theo thời gian

```
head(z)
```

```
           X.Low. X.High.
2000-07-28 100.00  100.00
2000-07-31 101.55  101.55
2000-08-02 103.38  103.38
2000-08-04 105.20  105.20
2000-08-07 106.92  106.92
2000-08-09 108.64  108.64
```

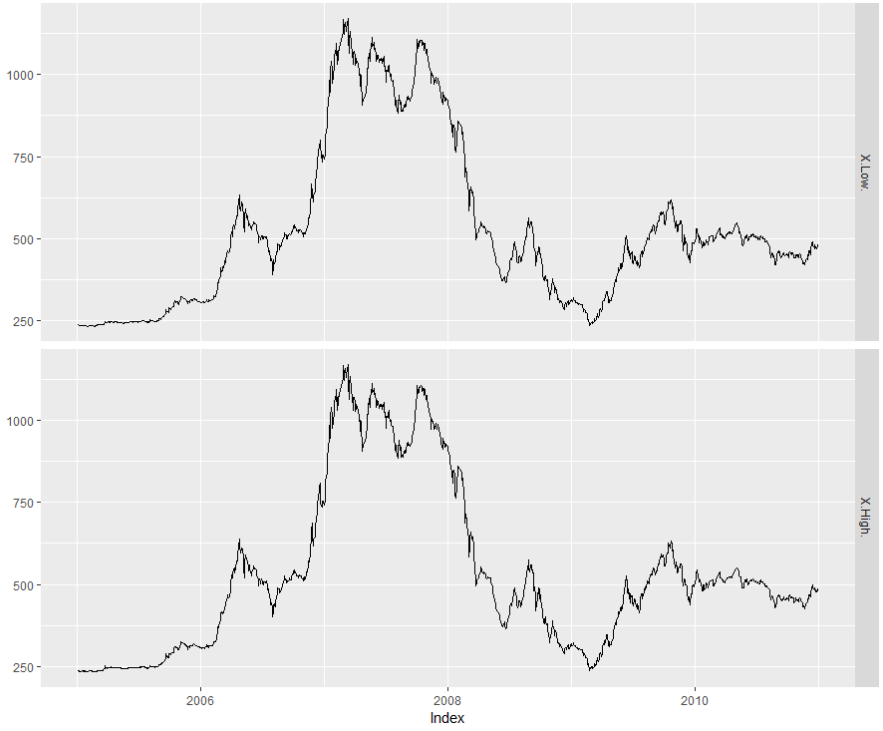
Vẽ biểu đồ VNIndex thấp nhất và cao nhất theo ngày

```
autoplot(z)
```



Nhìn vào dữ liệu giữa năm 2005 và 2010 thì có núi bất thường? Để xem chi tiết dữ liệu từ năm 2005 đến 2010 thì dùng lệnh window và vẽ biểu đồ:

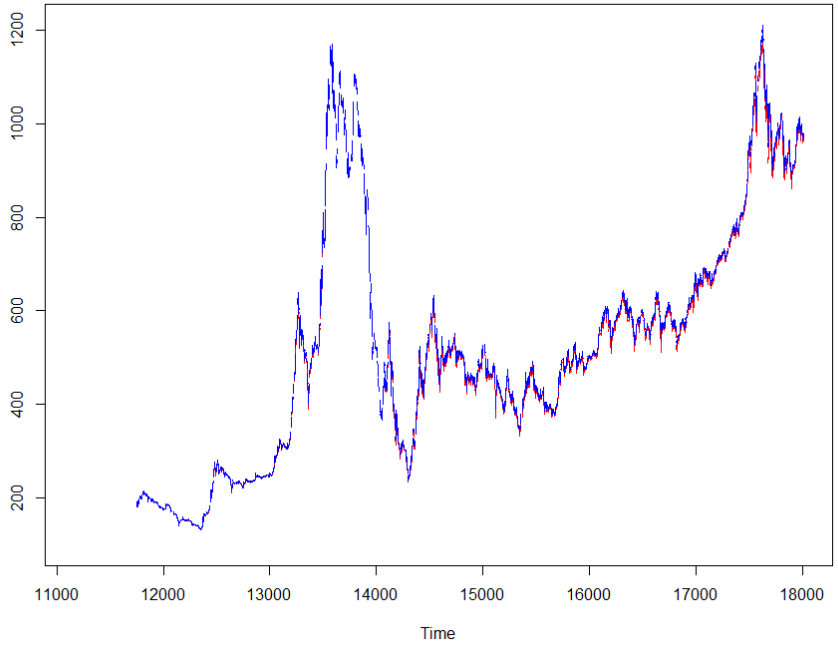
```
z1 = window(z, start = as.Date('2005/1/1'), end =
as.Date('2010/12/31'))

autoplot(z1)
```



Năm 2008 là năm khủng hoảng kinh tế nên chứng khoán lao dốc.

Xem giá trị Cao nhất và Thấp nhất trong cùng 1 biểu đồ

```
ts.plot(z, col = c("red", "blue"))
```



Sử dụng PerformanceAnalytics

```
install.packages('PerformanceAnalytics')
library('PerformanceAnalytics')
PerformanceAnalytics::chart.TimeSeries(z)
```

Gom dữ liệu theo tuần

Chúng ta cần bảng số liệu sau VNIndex High như sau:

	Mon	Tue	Wed	Thu	Fri
W1					
W2					
W3					

Sử dụng R để vẽ biểu đồ

Để xem được số liệu và biểu đồ liên quan đến chỉ số VN Index trước và sau lễ 30/4 thì cần lọc dữ liệu trước và sau ngày 30 tháng 4. Dữ liệu trong Bảng 1 ở trên có thể download tại:

"https://drive.google.com/open?id=1S3sf6YRT3Jt6a7U0n_I4mCCzKq6dqGkw"

Trong bài này dùng thư viện "ggplot2" để vẽ biểu đồ.

Trong R, dùng lệnh `install.packages(...)` để cài thư viện.

```
install.packages('ggplot2')
```

Các lệnh R sau đây sẽ xử lý một chút dữ liệu từ file csv và vẽ một số biểu đồ:

```
# Chọn file csv sau khi download
f = file.choose()
```

```
data = read.csv(f)


data$strDate = as.character(data$DTYYYYMMDD)
data$date = as.Date(data$strDate, format = '%Y%m%d')
# Xóa cột strDate
data$strDate = NULL
data$year = as.numeric(format(data$date, "%Y"))
data$month = as.factor(format(data$date, "%m"))
data$yyyymm = as.factor(format(data$date, "%Y-%m"))
data$m = as.numeric(format(data$date, "%m"))


library(ggplot2)
attach(data)
p = ggplot(data, aes(x = yyyymm, y = Close, fill = month))


p1 = p + geom_bar(stat="identity") + xlab("Ngày giao dịch trước và sau Lễ 30/4") + ylab("Giá đóng cửa")
p1 = p1 + theme(axis.text.x = element_text(angle = 90))
p1 = p1 + ggtitle("Giá đóng cửa chỉ số VN Index trước và sau lễ 30/4 trong 10 năm")
p1 = p1 + labs(fill = "Tháng")


plot(p1)


# Giá mở cửa
p = ggplot(data, aes(x = yyyymm, y = Open, fill = month))


p1 = p + geom_bar(stat="identity") + xlab("Ngày giao dịch trước và sau Lễ 30/4") + ylab("Giá mở cửa")
p1 = p1 + theme(axis.text.x = element_text(angle = 90))
p1 = p1 + ggtitle("Giá mở cửa chỉ số VN Index trước và sau lễ 30/4 trong 10 năm")
p1 = p1 + labs(fill = "Tháng")


plot(p1)



# Giá cho nhất
p = ggplot(data, aes(x = yyyymm, y = High, fill = month))
```

```
p1 = p + geom_bar(stat="identity") + xlab("Ngày giao dịch trước và sau Lễ 30/4") + ylab("Giá cao nhất")
p1 = p1 + theme(axis.text.x = element_text(angle = 90))
p1 = p1 + ggtitle("Giá cao nhất chỉ số VN Index trước và sau lễ 30/4 trong 10 năm")
p1 = p1 + labs(fill = "Tháng")


plot(p1)
```

Tham khảo

https://cran.r-project.org/web/packages/timeSeries/vignettes/timeSeriesPlot.pdf

Tp.HCM, ngày 1/5/2019

Quan sát giao dịch cổ phiếu VNM (Vinamilk)

Bài viết này minh họa bằng code Python.

Đọc dữ liệu

Để lấy dữ liệu thì tôi tự viết phần mềm để sưu tầm giao dịch theo lô của cổ phiếu VNM từ trang cafef.vn và lưu trên link:

https://thachln.github.io/datasets/VNM_20200710.zip.

Dữ liệu minh họa trong bài viết này được tập hợp từ ngày 10/27/2014 đến 10/7/2020.

```
import pandas as pd
df =
pd.read_csv('https://thachln.github.io/datasets/VNM_20200710.zip')
```

Hiểu một chút về dữ liệu

Dùng hàm info()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 674488 entries, 0 to 674487
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   symbol  674488 non-null  object
 1   time    674488 non-null  object
 2   price   674488 non-null  float64
 3   volume  674488 non-null  int64
dtypes: float64(1), int64(1), object(2)
```

Kết quả `info` cho thấy cột `time` có kiểu dữ liệu object chứ không phải là thời gian (datetime). Để chuyển kiểu cột time cho đúng kiểu thời gian thì sử dụng tiếp lệnh sau:

```
df['time'] = pd.to_datetime(df['time'])
```

Chạy lại lệnh info ở trên sẽ cho ra kết quả như sau:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 674488 entries, 0 to 674487
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   symbol  674488 non-null  object
 1   time    674488 non-null  datetime64[ns]
 2   price   674488 non-null  float64
 3   volume  674488 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(1), object(1)
```

Hàm `info()` cho thấy dataframe gồm 4 cột dữ liệu:

Cột	Ý nghĩa

symbol	là mã cổ phiếu. Trong dữ liệu này chỉ có cổ phiếu VNM
time	thời gian giao dịch. Kiểu dữ liệu là datetime64[ns]
price	giá giao dịch
volume	số lượng cổ phiếu được giao dịch

Dùng hàm describe() và thuộc tính shape

Xem vài thông tin thống kế về dữ liệu bằng hàm `describe()`:

```
df.describe()
```

```
                price         volume
count  674488.000000  6.744880e+05
mean      137.262721  1.565956e+03
std        28.859136  2.549325e+04
min        83.700000  1.000000e+00
25%       117.800000  8.000000e+01
50%       132.000000  3.200000e+02
75%       151.400000  1.040000e+03
max       215.000000  1.887654e+07
```

Xem thêm thuộc tính `shape`:

```
df.shape
```

```
(674488, 4)
```

Như vậy có thể tóm tắt vài chỉ số thống kê như sau:

- Dữ liệu có 674488 dòng, mỗi dòng là mỗi giao dịch.
- Giá trị mean (trung bình) cho thấy giá trung bình của cổ phiếu VNM là 138 nghìn đồng. Trung bình mỗi giao dịch $1.62 \times 10^3 = 1620$ cổ phiếu.
- Độ lệch chuẩn của giá cổ phiếu là 30. Chú ý giá cổ phiếu ở đây tính bằng đơn vị là **Nghìn** đồng. Điều này nói lên điều gì? Nó phản ánh sự khác biệt về giá trong các lần giao dịch. Các mức giá giao dịch có sự khác biệt nhau tầm 30 nghìn đồng xung quanh giá trung bình.

  Ở đây phải chú ý là chúng ta không có dữ liệu về các lần chia tách cổ phiếu. Mỗi lần chia tách thì giá cổ phiếu được điều chỉnh lại. Tạm thời bỏ qua yếu tố này để cho "bài tập thể dục" đơn giản.
- Tương tự, bạn có thể nhìn qua các chỉ số min; max; bách phân vị 25%, 50%, 75% của giá.

Xem vài dòng dữ liệu

```
df.head()
```

```
  symbol                 time  price  volume
```

```
0    VNM 2020-07-10 14:47:03  115.3   16660
1    VNM 2020-07-10 14:30:03  115.4    1000
2    VNM 2020-07-10 14:30:01  115.4     450
3    VNM 2020-07-10 14:29:45  115.4     660
4    VNM 2020-07-10 14:29:35  115.4     200
```

Thêm cột ngày

Hiện tại cột `time` chứa thời gian giao dịch đến mức giây. Các phân tích tiếp theo của chúng ta là tính theo ngày nên cần phải thêm cột date để chứa ngày tháng năm.

```
df['date'] = df['time'].dt.date

df.head()
```

```
  symbol                 time  price  volume        date
0    VNM 2020-07-10 14:47:03  115.3   16660  2020-07-10
1    VNM 2020-07-10 14:30:03  115.4    1000  2020-07-10
2    VNM 2020-07-10 14:30:01  115.4     450  2020-07-10
3    VNM 2020-07-10 14:29:45  115.4     660  2020-07-10
4    VNM 2020-07-10 14:29:35  115.4     200  2020-07-10
```

Tính tổng giá trị giao dịch

```
df['trade_value'] = df['price'] * df['volume']

df.head()
```

```
  symbol                 time  price  volume        date  trade_value
0    VNM 2020-07-10 14:47:03  115.3   16660  2020-07-10    1920898.0
1    VNM 2020-07-10 14:30:03  115.4    1000  2020-07-10     115400.0
2    VNM 2020-07-10 14:30:01  115.4     450  2020-07-10      51930.0
3    VNM 2020-07-10 14:29:45  115.4     660  2020-07-10      76164.0
4    VNM 2020-07-10 14:29:35  115.4     200  2020-07-10      23080.0
```

Tính giá trị trung bình của cổ phiếu

Giá trị trung bình trong ngày bằng cách tính tổng các giá trị giao dịch trong ngày. Sau đó chia cho tổng lượng giao dịch trong ngày. Kết quả lưu trong dataframe mới `df_avg_price`.

```
df_avg_price = df.groupby(['date'])['volume', 'trade_value'].sum()

df_avg_price['avg_price'] = df_avg_price['trade_value'] /
df_avg_price['volume']


df_avg_price.head()
```

```
            volume  trade_value   avg_price
date
2014-10-27   68880    7223620.0  104.872532
2014-10-29   27810    2892260.0  104.000719
2014-10-30   49530    5197670.0  104.939834
2014-11-03   20410    2155160.0  105.593337
2014-11-05  125160   13045320.0  104.229147
```

Xem thông tin của dataframe `df_avg_price`:

```
df_avg_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1318 entries, 2014-10-27 to 2020-07-10
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   volume       1318 non-null   int64
 1   trade_value  1318 non-null   float64
 2   avg_price    1318 non-null   float64
dtypes: float64(2), int64(1)
```

Bạn để ý thì thấy dataframe `df_avg_price` không có cột `date`.

Thêm cột gom nhóm vào dataframe

Để thêm cột làm tiêu chí gop nhóm (cột date) vào dataframe thì dùng hàm `reset_index()`:

```
df_avg_price = df_avg_price.reset_index()
```

Xem lại thông tin của dataframe `df_avg_price`:

```
df_avg_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1318 entries, 0 to 1317
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         1318 non-null   object
 1   volume       1318 non-null   int64
 2   trade_value  1318 non-null   float64
 3   avg_price    1318 non-null   float64
dtypes: float64(2), int64(1), object(1)
```

Một điểm chú ý là cột `date` có kiểu dữ liệu là `object`.

Chuyển kiểu object sang dạng date

Để chuyển cột date đang là object sang kiểu thời gian thì dùng hàm `pd.to_datetime(…)`:

```
df_avg_price['date'] = pd.to_datetime(df_avg_price['date'])
```

Xem lại thông tin:

```
df_avg_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1318 entries, 0 to 1317
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         1318 non-null   datetime64[ns]
 1   volume       1318 non-null   int64
 2   trade_value  1318 non-null   float64
 3   avg_price    1318 non-null   float64
dtypes: datetime64[ns](1), float64(2), int64(1)
```

Xem vài dòng dữ liệu:

```
df_avg_price.head()
```

```
        date  volume  trade_value   avg_price
0 2014-10-27   68880    7223620.0  104.872532
1 2014-10-29   27810    2892260.0  104.000719
2 2014-10-30   49530    5197670.0  104.939834
3 2014-11-03   20410    2155160.0  105.593337
4 2014-11-05  125160   13045320.0  104.229147
```

Sắp xếp lại dataframe theo ngày giảm dần

```
df_avg_price = df_avg_price.sort_values(by = ['date'],
ascending=False)

df_avg_price.head()
```

```
           date   volume  trade_value   avg_price
1317 2020-07-10   677180   78415689.0  115.797408
1316 2020-07-09  1560410  181141529.0  116.085855
1315 2020-07-08   551310   63805697.0  115.734699
1314 2020-07-07  1024960  119346196.0  116.439857
1313 2020-07-06  1319980  152240966.0  115.335813
```

Xem vài thông tin mô tả dataframe mới

```
df_avg_price.describe()
```

```
             volume   trade_value    avg_price
count  1.318000e+03  1.318000e+03  1318.000000
mean   8.013798e+05  1.112912e+08   136.770054
std    8.625206e+05  1.238176e+08    26.715244
min    2.913000e+03  3.612120e+05    84.764400
25%    3.382375e+05  4.410212e+07   119.588663
50%    6.499100e+05  8.893423e+07   133.040233
75%    1.060290e+06  1.503052e+08   148.547710
max    1.907396e+07  2.733192e+09   214.305513
```

Hãy quan sát vài chỉ số của cột bên trái đối với giá trung bình của cổ phiếu (cột `avg_price`)!

Tính chênh lệch giá giữa 2 ngày liền kề

```
df_avg_price['delta'] = df_avg_price['avg_price'].diff(periods = -1)

df_avg_price.head()
```

```
             volume  trade_value   avg_price     delta
date
2020-07-10   677180   78415689.0  115.797408 -0.288447
2020-07-09  1560410  181141529.0  116.085855  0.351156
2020-07-08   551310   63805697.0  115.734699 -0.705158
2020-07-07  1024960  119346196.0  116.439857  1.104045
2020-07-06  1319980  152240966.0  115.335813  1.370475
```

Thêm cột kí hiệu giá tăng hay giảm

Thêm cột pn (Positive or Negative) để ghi chú giá trung bình của cổ phiếu là tăng (1) hay giảm (-1) hay bằng (0) so với ngày hôm trước.

Giả định nếu giá tăng chưa tới 1 đồng thì xem như không tăng. Code bên dưới sẽ thêm côt pn với giá trị là 0 (xem như giá cổ phiếu không tăng so với ngày hôm trước.

Sau đó thiết lập lại giá trị nếu tăng trên 0.009 nghìn thì thiết lập cột `pn` là 1. Ngược lại nếu giảm hơn 0.009 nghìn thì thiết lập cột `pn` là -1.

```
df_avg_price['pn'] = 0
df_avg_price.loc[df_avg_price['delta'] > 0.009, 'pn'] = 1
df_avg_price.loc[df_avg_price['delta'] < -0.009, 'pn'] = -1
```

Xem thử kết quả

```
df_avg_price.head()
```

```
            volume  trade_value   avg_price     delta  pn
date
2020-07-10   677180   78415689.0  115.797408 -0.288447  -1
2020-07-09  1560410  181141529.0  116.085855  0.351156   1
2020-07-08   551310   63805697.0  115.734699 -0.705158  -1
2020-07-07  1024960  119346196.0  116.439857  1.104045   1
2020-07-06  1319980  152240966.0  115.335813  1.370475   1
```

Đến đây thì trong tay của bạn đã có dữ liệu giá trung bình cổ phiếu VNM mỗi ngày và cột `delta`, `pn` cho biết sự chêch lệch giá giữa hay ngày liền kề, cụ thể là tăng so với ngày hôm trước (cột `pn` có giá trị 1) hoặc giảm so với ngày hôm trước (cột `pn` có giá trị -1).

Xem lại các chỉ số thống kê của dataframe `df_avg_price`:

```
df_avg_price.describe()
```

```
             volume   trade_value    avg_price        delta           pn
count  1.318000e+03  1.318000e+03  1318.000000  1317.000000  1318.000000
mean   8.013798e+05  1.112912e+08   136.770054     0.008295    -0.015175
std    8.625206e+05  1.238176e+08    26.715244     2.653074     0.997223
min    2.913000e+03  3.612120e+05    84.764400   -29.834159    -1.000000
25%    3.382375e+05  4.410212e+07   119.588663    -0.871813    -1.000000
50%    6.499100e+05  8.893423e+07   133.040233    -0.020519    -1.000000
75%    1.060290e+06  1.503052e+08   148.547710     1.052080     1.000000
max    1.907396e+07  2.733192e+09   214.305513    21.134308     1.000000
```

Thống kê thử số ngày tăng, số ngày giảm

Thống kê thử số ngày tăng, số ngày giảm bằng hàm `crosstab(...)`:

```
pn_count = pd.crosstab(index=df_avg_price['pn'],  columns='count')
print(pn_count)
```

```
col_0  count
pn
-1       665
 0         8
 1       645
```

Số ngày tăng (645) không khác biệt lắm so với số ngày giảm (665).

Thêm cột ngày trong tuần

Thêm cột day cho 2 dataframe `df` và `df_avg_price`:

```
df['day'] = df['time'].dt.dayofweek
df_avg_price['day'] = df_avg_price['date'].dt.dayofweek
```

Xem dữ liệu của dataframe `df`:

```
df.head()
```

	symbol	time	price	volume	date	day	trade_value
0	VNM	2020-07-10 14:47:03	115.3	16660	2020-07-10	4	1920898.0
1	VNM	2020-07-10 14:30:03	115.4	1000	2020-07-10	4	115400.0
2	VNM	2020-07-10 14:30:01	115.4	450	2020-07-10	4	51930.0
3	VNM	2020-07-10 14:29:45	115.4	660	2020-07-10	4	76164.0
4	VNM	2020-07-10 14:29:35	115.4	200	2020-07-10	4	23080.0

Xem dữ liệu của dataframe `df_avg_price`:

```
df_avg_price.head()
```

	date	volume	trade_value	avg_price	delta	pn	day
1317	2020-07-10	677180	78415689.0	115.797408	-0.288447	-1	4
1316	2020-07-09	1560410	181141529.0	116.085855	0.351156	1	3
1315	2020-07-08	551310	63805697.0	115.734699	-0.705158	-1	2
1314	2020-07-07	1024960	119346196.0	116.439857	1.104045	1	1
1313	2020-07-06	1319980	152240966.0	115.335813	1.370475	1	0

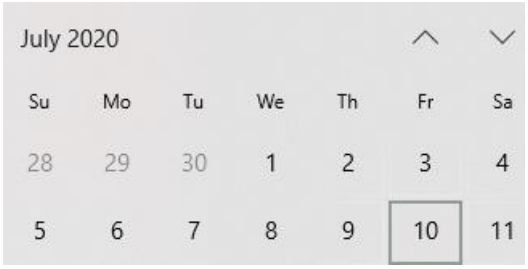
Đếm số lượng các thứ trong tuần

Sử dùng hàm `crosstab()`:

```
pd.crosstab(index=df['day'], columns='count')
```

col_0	count
day	
0	134510
1	138445
2	135880
3	128702
4	136951

Tra lịch ngày 10/7/2020 là thứ Sáu, thuộc tính `dayofweek` của cột `date` cho giá trị là 4 (cột `day`)



Hàm `crosstab(...)` ở trên cho thấy thứ trong tuần được đánh số từ 0 tới 4 tương ứng với Thứ Hai đến Thứ Sáu.

Thống kê số ngày tăng/giảm/không đổi theo thứ trong tuần

```
df_day_pn = pd.crosstab(df_avg_price['day'], df_avg_price['pn'])
```

```
df_day_pn
```

```
pn     -1   0    1
day
0     138   2  118
1     137   3  120
2     121   1  143
3     126   1  140
4     143   1  124
```

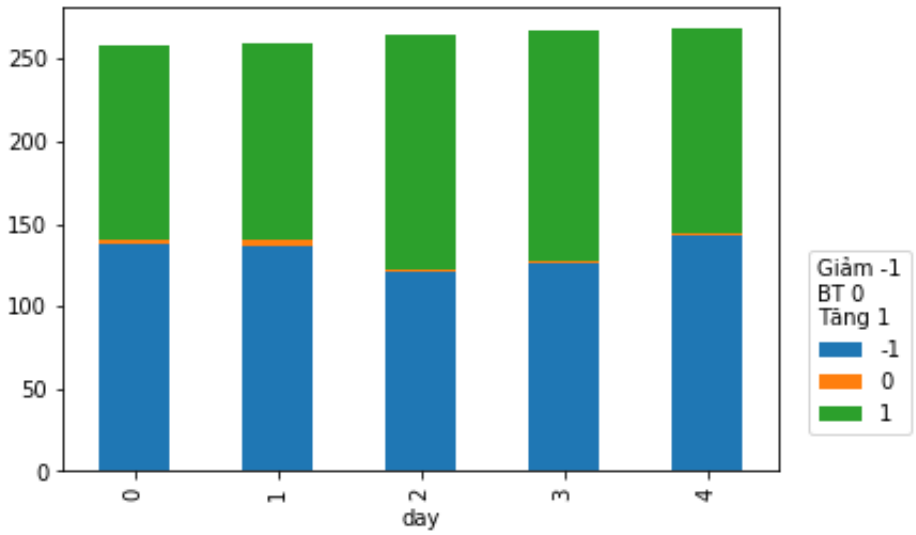
Kết quả cho thấy vài thông tin:

- Trong Thứ 2, số ngày giảm là 138, số ngày tăng là 118. Chú ý khái niệm Tăng/Giảm ở đây là so với ngày giao dịch trước đó (ở đây là Thứ Sáu tuần trước).
- Trong Thứ 3, số ngày giảm là 137, số ngày tăng là 120.
- Trong Thứ 4, số ngày giảm là 121, **số ngày tăng là 143**.
- Trong Thứ 5, số ngày giảm là 126, số ngày tăng là 140.
- Trong Thứ 6, **số ngày giảm là 143**, số ngày tăng là 124.

Theo số liệu thì Thứ 4 và Thứ 6 có vẻ ngược nhau. Thứ 4 thì tăng nhiều hơn so với giảm. Thứ 6 thì giảm nhiều hơn là tăng.

Vẽ biểu đồ

```
import matplotlib.pyplot as plt
df_day_pn.plot.bar(stacked=True)
plt.legend(title='Giảm -1\nBT 0\nTăng 1', bbox_to_anchor=(1.2, 0.5))
plt.show()
```



Câu hỏi đặt ra ở đây là số ngày tăng hoặc giảm trong một thứ nào đó có ý nghĩa thống kê hay không?

Ví dụ nhìn biểu đồ thì có vẻ ngày thứ 4 là tỉ số Tăng/Giảm nhiều nhất.

Xem lại dữ liệu ngày Thứ 4 (cột `day` = 2)

```
df_day_pn
```

```
pn    -1   0    1
day
0    138   2  118
1    137   3  120
2    121   1  143
3    126   1  140
4    143   1  124
```

Trong bộ dữ liệu có 121 + 1 + 143 = 265 ngày thứ Tư. Trong đó có 121 ngày giảm (chiếm 121/265 = **45.66%**) và 143 ngày tăng, chiếm 143/265 = **53.96%**)

Thử tính chỉ số ztest và p-value (Xem lại kiến thức Ngày 3, Bài 14: So sánh 2 tỉ lệ)

```
import numpy as np
from statsmodels.stats.proportion import proportions_ztest

# Count là số ngày tăng và số ngày giảm trong thứ 4
count = np.array([143, 121])
nobs = np.array([265, 265])
zstat, pval = proportions_ztest(count, nobs)
print('Tỉ số ztest:', zstat)
print('Trị số p:', pval)
```

```
Tỉ số ztest: 1.9112514762620285
Trị số p: 0.05597227155191926
```

Chỉ số ztest là 1.9, **gần 2 lần**. Chứng tỏ **có thể có sự khác biệt giữa số ngày tăng và số ngày giảm của cổ phiếu VNM trong Thứ 4**. Đồng thời trị số p = 0.056, hơi lớn hơn 0.05. Về lý thuyết diễn giải theo trị số p là không có ý nghĩa thống kê.

Tuy nhiên, giá trị ztest và trị số p đang rất sát ngưỡng "có ý nghĩa thống kê". Kết quả này rất đáng xem xét giả thuyết: **Vào ngày thứ 4 thì cổ phiếu VNM thường là tăng**. Nhìn vào biểu đồ sẽ suy đoán là ngày Thứ 5 đa số sẽ tăng. Sau đó đến ngày Thứ 6 thì đa số sẽ giảm.

Phần kiểm chứng giả thuyết trên thì nhường lại cho các chuyên gia về cổ phiếu nhé!

Nhìn bảng số liệu theo %

Thống kê theo % bằng cách sử dụng hàm `pd.crosstab(…)` với tham số `normalize`:

```
df_day_pn_percentage = pd.crosstab(df_avg_price['day'],
df_avg_price['pn'], normalize='index').round(4)*100

print('%Tăng/Giảm theo thứ:\n', df_day_pn_percentage)
```

```
%Tăng/Giảm theo thứ:
 pn      -1     0      1
day
0     53.49  0.78  45.74
1     52.69  1.15  46.15
2     45.66  0.38  53.96
3     47.19  0.37  52.43
4     53.36  0.37  46.27
```

Đọc và vẽ tín hiệu âm thanh

Bài viết này minh họa bằng code Python.

Tài liệu tham khảo chính:

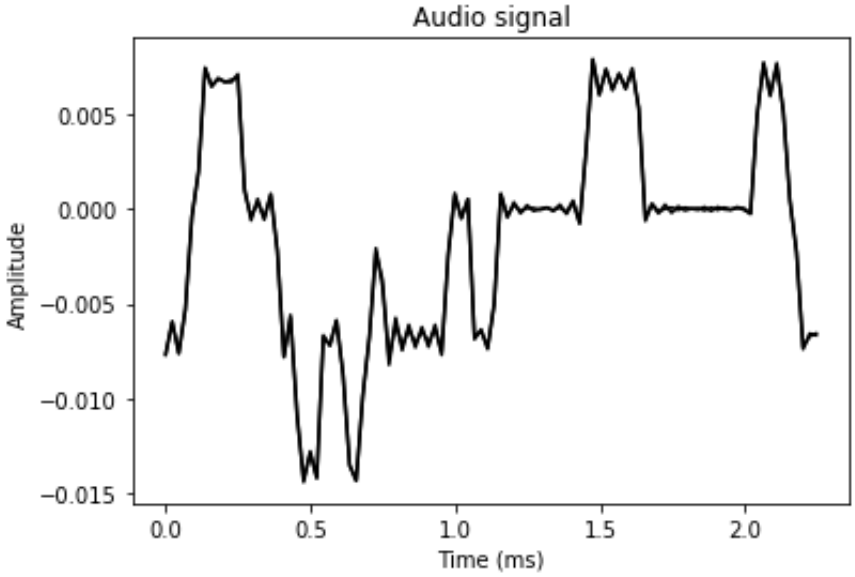
eBook "Python Machine Learning Cookbook", (Packt Publishing, 2019) của Prateek Joshi.

Tải file https://thachln.github.io/datasets/good-morning.wav về đường dẫn "D:/ai2020/data/good-morning.wav" và thực thi đoạn code sau:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

# Read the input file
wav_file_path = 'D:/ai2020/data/good-morning.wav'
sampling_freq, audio = wavfile.read(wav_file_path)

print('\nSampling frequence:', sampling_freq)

# Print the params
print('\nShape:', audio.shape)
print('Datatype:', audio.dtype)
print('Duration:', round(audio.shape[0] / float(sampling_freq), 3),
'seconds')



# Normalize the values
audio = audio / (2.**15)

# Extract first 100 values for plotting
audio = audio[0:100]



# Build the time axis
x_values = np.arange(0, len(audio), 1) / float(sampling_freq)

# Convert to seconds
x_values *= 1000
```

```
# Plotting the chopped audio signal
plt.plot(x_values, audio, color='black')
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.title('Audio signal')
plt.show()
```

```
Sampling frequence: 44100

Shape: (33897, 2)
Datatype: int16
Duration: 0.769 seconds
```



Một chút phân tích:

Bước 1: Import thư viện

Import các gói thư viện trong 3 dòng dầu tiên:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
```

Bước 2: Đọc file âm thanh và xem tần số mẫu

Đọc file âm thanh từ thư mục và hiển thị tần số mẫu.

```
# Read the input file
wav_file_path = 'D:/ai2020/data/good-morning.wav'
sampling_freq, audio = wavfile.read(wav_file_path)

print('\nSampling frequence:', sampling_freq)
```

```
Sampling frequence: 44100
```

Ghi chú:

File âm thanh (trong ví dụ này là file good-morning.wav là do tôi tự thu âm bằng phần mềm Audacity) là một file trên máy tính được lưu trữ dạng số (digitized version) của tín hiệu âm thanh thật (actual audio signals, trong tường hợp này là giọng nói "good morning" của tôi). Trong khi thu âm thì phần mềm Audacity mặc định cấu hình tần số mẫu (sample) là 44100 Hz. Tức là mỗi một giây phần mềm Audacity (kèm micro của máy tính) thu được 44100 phần tín hiệu âm thanh (audio parts). Nói cách khác mỗi phần tín hiệu này được lưu trong 1/44100 giây. Khi tần số lấy mẫu (sampling rate) cao thì chúng ta sẽ cảm giác tín hiệu âm thanh liên tục khi nghe lại bằng các thiết biệt phát âm thanh (audio players)

Bước 3: Xem tham số của âm thanh

Ba dòng tiếp theo hiển thị thêm các tham số của âm thanh

```
print('\nShape:', audio.shape)
print('Datatype:', audio.dtype)
print('Duration:', round(audio.shape[0] / float(sampling_freq), 3),
'seconds')
```

```
Shape: (33897, 2)
Datatype: int16
Duration: 0.769 seconds
```

Kết quả cho thấy có **2** luồng âm thanh, mỗi luồng có **33897** tín hiệu. Mỗi tín hiệu (audio signal) được lưu trong số nguyên có độ dài **16bit**.

Tính độ dài của đoạn âm thành bằng cách: lấy số tín hiệu nhân với 1/sampling_freq: 33897 * 1/44100 ≈ 0.769 **giây**.

Bước 4: Chuẩn hóa tín hiệu

Do tín hiệu âm thanh (audio signal) được lưu trong số nguyên có dấu với độ dài 16bit, nên chuẩn hóa bằng cách lấy độ lớn của tín hiệu chia cho 2^15

```
audio = audio / (2.**15)
```

Bước 5: Chọn tín hiệu để vẽ biểu đồ

Chọn 100 giá trị của tín hiệu đầu tiên:

```
audio = audio[0:100]
```

Tải sách nói "Từ tốt đến vĩ đại"

Mã nguồn Python sau đây sẽ giúp bạn tải các file audio sách nói "Từ Tốt Đến Vĩ Đại" từ trang web https://phatphapungdung.com/sach-noi/tu-tot-den-vi-dai-171248.html về thư mục 'D:/Temp/Tu-tot-den-vi-dai/'.

Hãy tự khám phá nội dung mã nguồn và sửa lại thư mục theo ý bạn nhé.

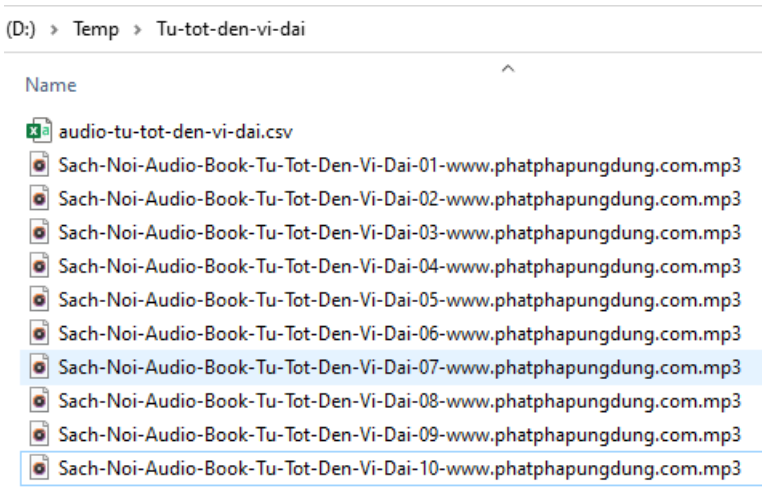
Gợi ý: Nên copy & paste mã nguồn vào phần mềm Spyder (xem lại Bài 4), chọn các dòng code và nhấn F9 để chạy các dòng đã chọn)

```
import bs4
from bs4 import BeautifulSoup
import json
import requests
import pandas as pd
import os
from pathlib import Path

# ========================================================================
# Download audio from <url> to <outFolder? with filename <audioName>
# ========================================================================
def downloadAudio(url, outFolder, audioName='null'):
    if (audioName == 'null'):
        audioName = Path(url).name
    fileOut = Path(os.path.join(outFolder, audioName))
    response = requests.get(url)

    fileOut.write_bytes(response.content)


    return
# ========================================================================
# Get list of audio link and title from website
# @return dataframe(title, url)
# ========================================================================
def parseData():
    titles = []
    urls = []
    url = 'https://phatphapungdung.com/sach-noi/tu-tot-den-vi-dai-
171248.html'
```

```
    response = requests.get(url)


    html_soup = BeautifulSoup(response.text, 'html.parser')


    html_data = html_soup.find('div', {'class': 'fp-playlist-
external'})


    # article_containers = html_soup.find_all("div", class_="fp-
playlist-external is-audio")


    # content = article_containers.text


    # print(html_data)


    for e in html_data:
        if isinstance(e, bs4.element.Tag):
            print(type(e))
            print('Element:', e)
            data_item_json = e['data-item']
            print('Type of data item', type(data_item_json))
            json_data = json.loads(data_item_json)
            print('Sources:', json_data['sources'])
            print('Title', json_data['fv_title'])


            title = json_data['fv_title']
            print("Type of json_data[sources]:",
type(json_data['sources']))


            print("Type of json_data[sources][0]:",
type(json_data['sources'][0]))


            dict_data = json_data['sources'][0]
            print(dict_data['src'])


            url = dict_data['src']


            print('Parsed data: (Title, Url)=', title, url)
            titles.append(title)
            urls.append(url)
```

```
        else:
            print('No parse:',type(e))
    return pd.DataFrame({'title': titles, 'url': urls})



# Change your folder to contains books
outFolder = 'D:/Temp/Tu-tot-den-vi-dai/'
# Create root folder
if (not os.path.exists(outFolder)):
    os.mkdir(outFolder)

df = parseData()

# Write dataframe into CSV
df.to_csv(os.path.join(outFolder, 'audio-tu-tot-den-vi-dai.csv'),
encoding='utf-8')

# Scan all data frame of free books
for index, row in df.iterrows():
    url = row['url']
    title = row['title']
    print('...')
    downloadAudio(url, outFolder)
```

Kết quả được thư mục như sau:



Vẽ bản đồ Việt Nam

Cài thư viện

```
install.packages('raster')
```

Lấy dữ liệu bản đồ Việt Name

```
# Lấy dữ liệu cho VN ở cấp tỉnh:
library(raster)
vietnam = getData('GADM', country = 'Vietnam', level = 1)
head(vietnam)
```

```
   GID_0  NAME_0   GID_1                NAME_1 VARNAME_1 NL_NAME_1       TYPE_1
1    VNM Vietnam VNM.1_1             An Giang  An Giang       <NA> T<U+1EC9>nh
12   VNM Vietnam VNM.2_1       B<U+1EA1>c Liêu  Bac Lieu       <NA> T<U+1EC9>nh
23   VNM Vietnam VNM.3_1     B<U+1EAF>c Giang Bac Giang       <NA> T<U+1EC9>nh
34   VNM Vietnam VNM.4_1 B<U+1EAF>c K<U+1EA1>n   Bac Kan       <NA> T<U+1EC9>nh
45   VNM Vietnam VNM.5_1       B<U+1EAF>c Ninh  Bac Ninh       <NA> T<U+1EC9>nh
56   VNM Vietnam VNM.6_1        B<U+1EBF>n Tre   Ben Tre       <NA> T<U+1EC9>nh
   ENGTYPE_1 CC_1 HASC_1
1   Province <NA>  VN.AG
12  Province <NA>  VN.BL
23  Province <NA>  VN.BG
34  Province <NA>  VN.BK
45  Province <NA>  VN.BN
56  Province <NA>  VN.BR
```

Xem các cột dữ liệu

```
names(vietnam)
```

```
[1] "GID_0"     "NAME_0"    "GID_1"     "NAME_1"    "VARNAME_1" "NL_NAME_1" "TY
PE_1"
 [8] "ENGTYPE_1" "CC_1"      "HASC_1"
```

Xem kiểu dữ liệu của biến vietnam

```
class(vietnam)
```

```
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

Plot bản đồ

```
plot(vietnam)
```

Ghi chú: Có vẻ dữ liệu về Quần đảo Hoàng Sa và Trường Sa của Việt Nam không rõ ràng.

Code đầy đủ

```
# Lấy dữ liệu cho VN ở cấp tỉnh:
library(raster)
vietnam = getData('GADM', country = 'Vietnam', level = 1)
plot(vietnam)
```

Đọc ảnh y khoa DiCOM

Đoạn code code sẽ đọc ảnh DiCOM từ thư mục, hiển thị vài thông tin cơ bản và hiển thị ảnh:

Tham khảo code:

https://pydicom.github.io/pydicom/stable/auto_examples/input_output/plot_read_dicom.html

```
import matplotlib.pyplot as plt
import pydicom
filePath = 'D:/ai2020/data/mri/ThachLN.dcm'
dataset = pydicom.dcmread(filePath)



print("Storage type.....:", dataset.SOPClassUID)

pat_name = dataset.PatientName
display_name = pat_name.family_name + ", " + pat_name.given_name
print("Patient's name...:", display_name)
print("Patient id.......:", dataset.PatientID)
print("Modality.........:", dataset.Modality)
print("Study Date.......:", dataset.StudyDate)

if 'PixelData' in dataset:
    rows = int(dataset.Rows)
    cols = int(dataset.Columns)
    print("Image size.......: {rows:d} x {cols:d}, {size:d}
bytes".format(
        rows=rows, cols=cols, size=len(dataset.PixelData)))
    if 'PixelSpacing' in dataset:
        print("Pixel spacing....:", dataset.PixelSpacing)

# use .get() if not sure the item exists, and want a default value if
missing
print("Slice location...:", dataset.get('SliceLocation', "(missing)"))

# plot the image using matplotlib
plt.imshow(dataset.pixel_array, cmap=plt.cm.bone)
plt.show()
```

```
Storage type.....: 1.2.840.10008.5.1.4.1.1.4
Patient's name...: Le Ngoc Thach,
Patient id.......: ThachLN.github.io
Modality.........: MR
Study Date.......: 20200608
Image size.......: 512 x 512, 524288 bytes
Pixel spacing....: [0.3515625, 0.3515625]
Slice location...: 25.713560265779
```