FloatingPointError F •••

```
!pip install pandas-datareader statsmodels
```

```
Requirement already satisfied: pandas-datareac
Requirement already satisfied: statsmodels in
Requirement already satisfied: lxml in /usr/lo
Requirement already satisfied: pandas>=0.23 ir
Requirement already satisfied: requests>=2.19.
Requirement already satisfied: numpy<3,>=1.22.
Requirement already satisfied: scipy!=1.9.2,>=
Requirement already satisfied: patsy>=0.5.6 ir
Requirement already satisfied: packaging>=21.3
Requirement already satisfied: python-dateutil
Requirement already satisfied: pytz>=2020.1 ir
Requirement already satisfied: tzdata>=2022.7
Requirement already satisfied: charset-normali
Requirement already satisfied: idna<4,>=2.5 ir
Requirement already satisfied: urllib3<3,>=1.2
Requirement already satisfied: certifi>=2017.4
Requirement already satisfied: six>=1.5 in /us
```

```
import pandas as pd
import pandas_datareader.data as web
import statsmodels.formula.api as sm
```

```
# Download Fama-French 3-factor data (monthly)
ff_data = web.DataReader('F-F_Research_Data_Factor
ff_data = ff_data[0]  # Get the monthly data (the
ff_data.index = ff_data.index.to_timestamp() # Cor
ff_data = ff_data / 100 # Convert percentages to d
```

```
<ipython-input-18-a39eb10acee1>:2: FutureWarni
  ff_data = web.DataReader('F-F_Research_Data_
<ipython-input-18-a39eb10acee1>:2: FutureWarni
  ff_data = web.DataReader('F-F_Research_Data_
```

```
import yfinance as yf
import pandas as pd

# Assuming you want to analyze Apple's stock retu
ticker = "AAPL"
start_date = '2020-01-01'

# Download stock data using yfinance
stock_data = yf.download(ticker, start=start_date)

# Extract the adjusted closing prices, which are c
# Changed 'Adj Close' to 'Close' as this is the co
stock_returns = stock_data['Close']

# Now you can proceed with the merge
df = pd.merge(stock_returns, ff_data, left_index=T
df.rename(columns={'Close': 'stock_returns'}, inpl
```

```
[*********************100%**********************
```

```
import yfinance as yf

end_date = datetime.date.today().strftime('%Y-%m-%d
apple = yf.Ticker("AAPL")
AAPL = apple.history(start = "2020-01-01", end= end
AAPL.head()
```

Date	Open	High	Low	Close
2020-01-02 00:00:00-05:00	71.721011	72.776591	71.466805	72.71606
2020-01-03 00:00:00-05:00	71.941321	72.771737	71.783954	72.00910
2020-01-06 00:00:00-05:00	71.127873	72.621654	70.876083	72.58291

Next steps: Generate code with AAPL View recommended plo

Get the Balance Sheet and Income Statements

```
balance_sheet = apple.balance_sheet
print("Balance Sheet:")
print(balance_sheet.head())

income_statement = apple.financials
print("\nIncome Statement:")
print(income_statement.head())

# Information about Apple:
info = apple.info
print(f"\nCompany: {info['longName']}")
```

```
print(f"Company: {info['longName']}")
print(f"Sector: {info['sector']}")
print(f"Industry: {info['industry']}")
print(f"Market Cap: {info['marketCap']}")
print(f"P/E Ratio: {info['trailingPE']}")

# dividend data
dividends = apple.dividends
print("Dividends:")
print(dividends.tail())
```

```
Balance Sheet:
                              2024-09-30      20
Treasury Shares Number               NaN
Ordinary Shares Number    15116786000.0   15550
Share Issued              15116786000.0   15550
Net Debt                  76686000000.0   81123
Total Debt               106629000000.0  111088

                              2021-09-30 2020-09
Treasury Shares Number               NaN
Ordinary Shares Number    16426786000.0
Share Issued              16426786000.0
Net Debt                  89779000000.0
Total Debt               136522000000.0

Income Statement:

Tax Effect Of Unusual Items
Tax Rate For Calcs
Normalized EBITDA
Net Income From Continuing Operation Net Minor
Reconciled Depreciation


Tax Effect Of Unusual Items
Tax Rate For Calcs
Normalized EBITDA
Net Income From Continuing Operation Net Minor
Reconciled Depreciation


Tax Effect Of Unusual Items
Tax Rate For Calcs
Normalized EBITDA
Net Income From Continuing Operation Net Minor
Reconciled Depreciation


Tax Effect Of Unusual Items
```

```
Tax Rate For Calcs
Normalized EBITDA
Net Income From Continuing Operation Net Minor
Reconciled Depreciation

Company: Apple Inc.
Sector: Technology
Industry: Consumer Electronics
Market Cap: 2590110646272
P/E Ratio: 27.368254
Dividends:
Date
2024-02-09 00:00:00-05:00    0.24
2024-05-10 00:00:00-04:00    0.25
2024-08-12 00:00:00-04:00    0.25
2024-11-08 00:00:00-05:00    0.25
2025-02-10 00:00:00-05:00    0.25
Name: Dividends, dtype: float64
```

```
apple = yf.Ticker("AAPL")

tickers = ["SPY", "AAL", "ZM", "NFLX", "META", 'AA

end_date = datetime.date.today().strftime('%Y-%m-%
apple = yf.Ticker("AAPL")
AAPL = apple.history(start = "2020-01-01", end= "2

for ticker in tickers:
    globals()[ticker] = yf.Ticker(ticker)
    globals()[ticker] = globals()[ticker].history(
```

```
print(META.Close.mean())
META.describe()
```

```
299.9754145219389
```

	Open	High	Low	
count	1257.000000	1257.000000	1257.000000	1257.0
mean	299.811910	304.029698	295.797743	299.9
std	124.745251	125.702634	123.419958	124.6
min	89.657445	90.035660	87.676766	88.4
25%	207.860343	210.607414	205.541276	208.7
50%	277.850522	283.892024	274.984004	279.5
75%	345.003997	350.448357	341.570169	344.6
max	630.430194	637.318496	626.147483	631.6

Now, let us keep only the closing prices for our analysis.

```
## keep only column close for all tickers
for ticker in tickers:
    globals()[ticker] = globals()[ticker].Close

SPY
```

Date	Close
2020-01-02 00:00:00-05:00	300.291504
2020-01-03 00:00:00-05:00	298.017792
2020-01-06 00:00:00-05:00	299.154602
2020-01-07 00:00:00-05:00	298.313507
2020-01-08 00:00:00-05:00	299.903351
...	...
2024-12-23 00:00:00-05:00	592.906433
2024-12-24 00:00:00-05:00	599.496582
2024-12-26 00:00:00-05:00	599.536499
2024-12-27 00:00:00-05:00	593.225464
2024-12-30 00:00:00-05:00	586.455811

1257 rows × 1 columns

dtype: float64

```
## keep only column close for all tickers
for ticker in tickers:
    # Assign the 'Close' column to the ticker var
    globals()[ticker] = globals()[ticker]['Close']

SPY
```

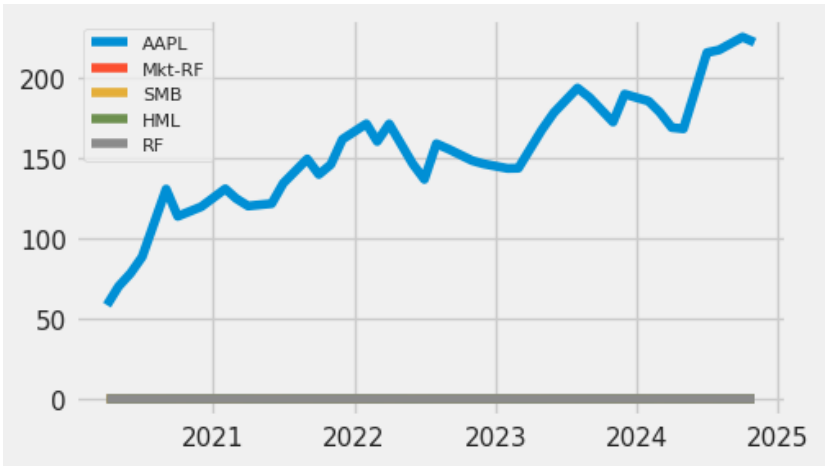
Date	Close
2020-01-02 00:00:00-05:00	300.291504
2020-01-03 00:00:00-05:00	298.017700
2020-01-06 00:00:00-05:00	299.154633
2020-01-07 00:00:00-05:00	298.313507
2020-01-08 00:00:00-05:00	299.903381
...	...
2024-12-23 00:00:00-05:00	592.906433
2024-12-24 00:00:00-05:00	599.496582
2024-12-26 00:00:00-05:00	599.536499
2024-12-27 00:00:00-05:00	593.225464
2024-12-30 00:00:00-05:00	586.455811

1257 rows × 1 columns

dtype: float64

```
# print(AAPL)
# print(AAPL.shift(1))
```

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5, 3))
plt.plot(df, label=df.columns)
plt.legend(loc='upper left',fontsize=8)
plt.show()
```



For financial analysis, we require the log returns (daily), rather than the raw stock prices. The formula for log returns is:

log(Today's Price/yesterday's price - 1)

Find the betas of the stocks. The formula is shown below:

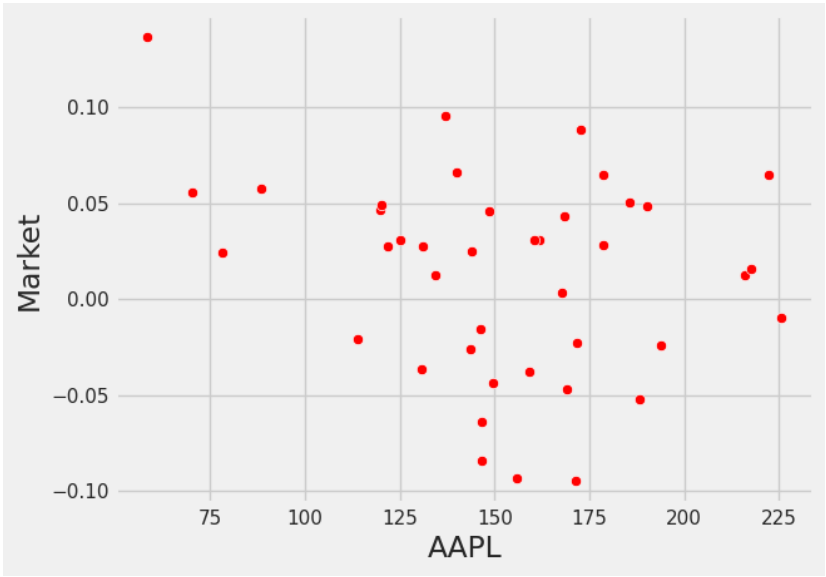
```
beta_aapl = (df[['Mkt-RF','AAPL']].cov()/df['Mkt-R
```

Calculate beta using regression line.

```
import numpy as np
import pandas as pd  # Import pandas if not alread

# Assuming df contains your stock and market data
# If not, adjust this line to use the correct Data
data = df[['Mkt-RF', 'AAPL']]

# Rename the columns to 'Market' and 'AAPL' to mat
data = data.rename(columns={'Mkt-RF': 'Market', 'A

beta, alpha = np.polyfit(data['Market'], data['AAP
alpha
beta
```

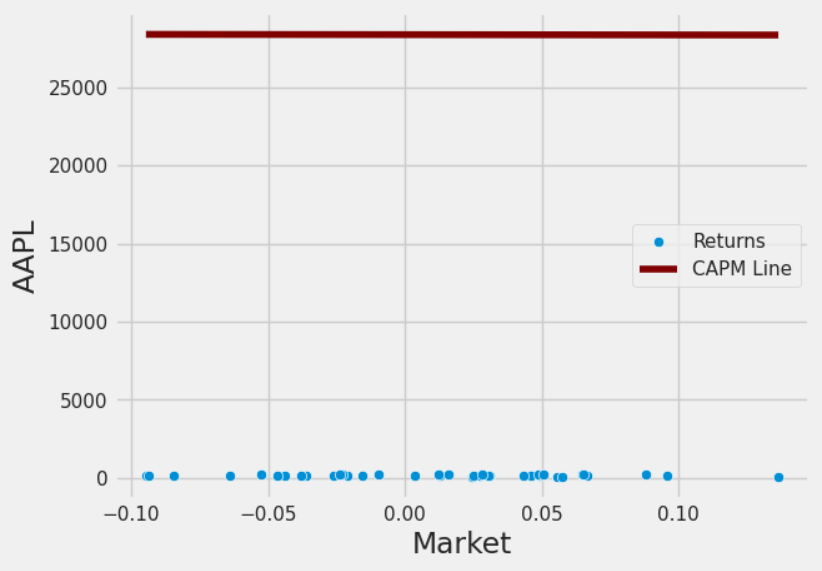
```
np.float64(-182.46005968691617)
```

```
# plt.axvline(0, color='grey', alpha = 0.5)
# plt.axhline(0, color='grey', alpha = 0.5)
sns.scatterplot(y = 'Market', x = 'AAPL', data = d
plt.show()
```



```
sns.scatterplot(y = 'AAPL', x = 'Market', data = d
sns.lineplot(x = data['Market'], y = alpha + (data
```

```
<Axes: xlabel='Market', ylabel='AAPL'>
```



Convert Daily Stock Market Returns to Annualized Returns (assuming 252 trading days in a year).

```
rm = data['Market'].mean()*252
rm
cov = data[['Market','AAPL']].cov() *252
cov_aapl_market = cov.iloc[0,1]
cov_aapl_market
market_var = data['Market'].var()*252
market_var

AAPL_beta_annual = cov_aapl_market / market_var
print('The annualized beta will equal the one calc

rf = 0.025
riskpremium = rm - rf

## CAPM
AAPL_capm_return = rf + AAPL_beta_annual*riskprem

print(f"The annualized CAPM return of AAPL is {AAF
```

```
The annualized beta will equal the one calcula
The annualized CAPM return of AAPL is -57061.1
```

```
sharperatio = (rm-rf)/(data['AAPL'].std()*np.sqrt(
sharperatio
print(f"Sharpe Ratio: {round(sharperatio,4)}")
```

```
Sharpe Ratio: 0.0051
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Analyze your files with code written by Gemini

Upload

Enter a prompt here

0 / 2000

Gemini can make mistakes so double-check responses and use code with caution. Learn more