University of St.Gallen

Fundamental Stock Screener

May 26, 2019

University of St. Gallen

Skills: Programming with Advanced Computer Languages

Group ID: 1106

Authors:	Student-ID:
Till Folger	14-617-765
Verena Konzett	18-613-315
Andreas Kröbl	17-601-014

Contents

1 Task and programming procedure

The purpose of the program is to identify stocks which appear to be fundamentally undervalued. The program scans the S&P 500 or the Dow Jones Industrial Index for stocks which meet the user's criteria. A common ratio to screen for is the Price to Book ratio, which compares the price of a stock to the book value of the company's assets per share. Our program enables the user to screen for three fundamental ratios: Price to Book, Price to Earnings and Debt to Equity. After the screening process the user has the option to print the identified stocks with their respective ratios to an Excel File for future use.

For a solid investment decision, a peer group benchmarking is essential. Therefore, we have implemented a peer benchmarking with the most popular valuation benchmarks. The user can either add peer stocks to the benchmarking by manually entering stock tickers or by using peers, which are recommended by the program itself. By looking at the mean and median of the peer valuation multiples the user can quickly get a picture whether the stock is overpriced or underpriced.

In order to get further insight, the program creates a visual comparison of the stock price movement compared to a selected industry index and its peers. As an additional feature the user can investigate the stock price development and compare it to two moving averages. Finally, two plots targeting volatility shall give the user an understanding of the historic volatility and the rolling volatility. With the help of the program the user can quickly screen through hundreds of stocks and can get a first understanding of a potential investment target.

2 Code

We recommend using Spyder to execute the code.

Preamble:
Necessary PIPs in Anaconda Prompt:

Pandas:	pip install pandas
Numpy:	pip install numpy
Request	pip install request
Matplotlib:	pip install matplotlib
Beautiful Soup:	pip install beautifulsoup4
Selenium:	pip install selenium
Datetime	pip install datetime
Time	pip install time

The package 'Selenium' requires the user to install the latest google chrome driver and to specify the executable path. See line 16.

Link to download chrome driver: http://chromedriver.chromium.org/downloads

The function print_full() was created in order to display the whole set of possible peer companies, which the user can access by typing in help when adding peer companies. Without *print_full* only 60 companies instead of 500 would be shown.

```
1.  # Load libraries
2.  import pandas as pd
3.  from pandas_datareader import data as pdr
4.  import numpy as np
5.  import datetime as dt # package for dates
6.  import time
7.  import matplotlib.pyplot as plt # for graphs
8.
9.
10. # packages for scraping:
11. import bs4 as bs # scraping
12. import requests # get request
13. from selenium import webdriver # scraping
14. from selenium.common.exceptions import NoSuchElementException # error handling of scrap
    ing
15.
16. # set executable_path
17. executable_path=r"C:\Users\Andi\Desktop\test2\chromedriver.exe"
18. #executable_path=r"C:\Users\Till\Desktop\test\chromedriver_win32\chromedriver.exe"
19.
20. #define a function to print whole data set
21. def print_full(x) -> None:
22.     pd.options.display.max_columns = x.shape[1] # display all columns
23.     pd.options.display.max_rows = x.shape[0] # display all rows
24.     print(x)
```

Step 1:

As a first step the indexes are defined, and its constituents are included in an array for each index. The test list has the purpose to efficiently check the code without screening too many companies. With *pd.read_excel('Ticker_List.xlsx')* we access an Excel file, which includes all possible tickers and respective company names.

```
1.  ################################
2.  ## STEP 1: PROVIDE STOCK INDICES
3.  ################################
4.
5.  '''''
6.  We provide two current stock indices with tickers and a short test list where you can c
    hoose from.
7.  Additionally, we provide a ticker list with company names from which the user can choos
    e in STEP 3.
8.  '''
9.
10. # Current S&P Companies
11. SP500 = ['A', 'AAL', 'AAP', 'AAPL', 'ABBV', 'ABC', 'ABMD', 'ABT', 'ACN', 'ADBE'…]
```

```
12.
13. # Current Dow Jones Industrial Companies
14. DJI = ['MMM', 'AXP', 'AAPL', 'CAT', 'CVX',  'CSCO', 'DWDP', 'XOM', 'INTC', 'IBM', …]
15.
16. # test list
17. test = ['GT', 'IVZ', 'LNC', 'MET', 'MOS', 'MPC', 'MRO', 'PRU', 'TAP', 'UNM', 'WRK']
18.
19. # load ticker data
20. Ticker_List = pd.read_excel('Ticker_List.xlsx')
```

Step 2:

In a second step the user selects an index and inputs all the relevant key metrics. The program scrapes the required information from yahoo finance. We use the function *soup.find_all* from beautiful soup as it works well for html codes and is fast. The user further can save the results from the scrape into an excel file called "MeetRequirements".

```
1.  ###############################
2.  # STEP 2: STOCK SCREENER
3.  ###############################
4.
5.  ###############################
6.  # STEP 2.1 ASK USER FOR INPUT
7.  ###############################
8.
9.  # print welcome message
10. print("\nWelcome to the Fundamental Stock Screener!\nThis tool helps you to efficiently
     filter through stocks for your fundamental stock analysis and stock picking!")
11.
12. # Ask user for index
13. print("\nYou can choose between two stock indices: Dow Jones Industrial (DJI) and S&P50
    0. Which one do you want to choose?")
14.
15. # Check for user input
16. while True:
17.     try:
18.         # first get index from user:
19.         index = input(("Please enter either DJI or SP500 here: "))
20.         # check for input
21.         if index.lower() == "dji": #check if input is equal to index dji
22.             # use DJI later
23.             usedindex = DJI
24.             break # everything ends
25.         elif index.lower() == "sp500": #check if input is equal to index sp500
26.             # use SP500 later
27.             usedindex = SP500
28.             break # everthing ends
29.         elif index.lower() == "test": #check if input is equal to index test
30.             # use test later
31.             usedindex = test
32.             break
33.         #otherwise, give user again chance for input
34.         else:
35.             print("\nError. You must insert exactly one of the indices - DJI or SP500 o
    r test - given. Please try again.") # print error message if input is not a number
36.             continue # ask user again to give an input
37.
38.
```

```python
39.      # brauchen wir das überhaupt? (ist bei den anderen loops gleich...)
40.      # any other error:
41.      except:
42.          print("\nError. You must insert exactly one of the indices - DJI or SP500 or te
    st - given. Please try again.") # print error message if input is not a number
43.          continue # ask user again to give an input
44.
45. print("\nThe stock index " + index + " will be analysed now. You now have to set the mi
    nimum requirements a stock must meet.")
46.
47.
48. # lists to save data:
49. statistics_names = []
50. statistics_values = []
51.
52. # ask user for maximum pbr
53. print("\nWhat is the maximum Price to Book Ratio a stock may have? Note that investors
    usually set a number around 2 here. ")
54. while True:
55.      try:
56.          # let user insert a number
57.          max_pbr = float(input(("Please insert the maximum number of the Price to Book R
    atio here: ")))
58.          # Append ratio name and value to lists:
59.          statistics_values.append(max_pbr)
60.          statistics_names.append(' Maximum Price to Book Ratio')
61.          break
62.      except ValueError:
63.          print("\nError. You have to enter a number. Make sure to use a dot to separate
    decimals. Please try again.") # print error message if input is not a number
64.          continue # ask user again to give an input which size will be checked again
65.
66.
67. # ask user for maximum p/e
68. print("\nWhat is the maximum Price to Earnings Ratio a stock may have? Note that invest
    ors usually set here a number around 20.")
69. while True:
70.      try:
71.          # let user insert a number for ratio
72.          max_pe = float(input(("Please insert the maximum number of the Price to Earning
    s Ratio here: ")))
73.          # Append ratio name and value to lists
74.          statistics_values.append(max_pe)
75.          statistics_names.append('Maximum Price to Earnings Ratio')
76.          break
77.      except ValueError:
78.          print("\nError. You have to enter a number. Please try again.") # print error m
    essage if input is not a number
79.          continue # ask user again to give an input which will be checked again
80.
81.
82. # ask user for maximum d/e
83. print("\nWhat is the maximum Debt to Equity Ratio a stock may have? Set it around 100 t
    o get a result.")
84. while True:
85.      try:
86.          # let user insert a number for ratio
87.          max_de = float(input(("Please insert the maximum number of the Debt to Equity R
    atio here: ")))
88.          # Append ratio name and value to lists
89.          statistics_values.append(max_de)
90.          statistics_names.append('Maximum Debt to Equity Ratio')
```

```python
91.          break
92.      except ValueError:
93.          print("\nError. You have to enter a number. Please try again.") # print error m
    essage if input is not a number
94.          continue # ask user again to give an input which size will be checked again
95.
96. # print message what follows
97. print("\nGreat! You will now get the stocks that meet the given requirements: \n")
98.
99.
100.        Statistics_Frame = pd.DataFrame({
101.                    'Ratio Name': statistics_names,
102.                    'Ratio Value': statistics_values})
103.
104.        print(Statistics_Frame)
105.
106.        print("\nThis process may take a few minutes. Please be patient.")
107.
108.
109.        ####################################################
110.        ## STEP 2.2: SCRAPE DATA AND CHECK FOR REQUIREMENTS
111.        ####################################################
112.
113.        # Ouput: List of stocks that meet requirements
114.
115.        # create empty lists to save results in function
116.        meetrequirements = []
117.        meetrequirements_stock_name = []
118.        PBR_list = []
119.        PE_list = []
120.        DE_list = []
121.        PEG_list = []
122.        error_list = []
123.
124.        #Webscrapper using Yahoo HTML Code / Here we use BeautifulSoup as it is faster t
    han Selenium and works accurately for this data
125.        def yahooKeystats(stock):
126.            try:
127.                # url to be visited for each stock
128.                url = "https://finance.yahoo.com/quote/" + stock + "/key-statistics"
129.                # send get request for url and parse text
130.                r=requests.get(url)
131.                soup = bs.BeautifulSoup(r.text,"html.parser")
132.                pbr = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[6].text
133.                stock_name = soup.find_all('h1',{'class':'D(ib) Fz(16px) Lh(18px)'})[0].
    text
134.
135.                # Check whether stocks meet some investment requirements set by user
136.
137.                # 1.) Price to Book Ratio must be smaller than 1
138.                if float(pbr) < float(max_pbr):
139.                    #Used PE not PEG as Yahoo doesn't have this for a lot of stocks = pr
    oducing errors
140.                    # then scrape p/e:
141.                    PE = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[2].text
142.
143.                    # 2.) PE ratio must be smaller than some value
144.                    if float(PE) < float(max_pe):
145.                        # then scrape debt to equity ratio:
146.                        DE = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[26].t
    ext
147.
```

```python
148.                     # 3.) DE ratio must be smaller than some value
149.                     if float(DE) < float(max_de):
150.                         PEG5 = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})
    [4].text
151.
152.                         # Output given to the user:
153.                         print(stock_name + ' meets the requirements:')
154.                         print('----------------------------------------')
155.                         print('Price to Book Ratio: ', pbr)
156.                         print('Price to Earnings Ratio: ', PE)
157.                         print('Total Debt to Equity: ', DE)
158.                         print('PEG Ratio: ', PEG5)
159.                         print('----------------------------------------')
160.
161.                         # Append ratios and the ticker of a stock to lists created a
    bove
162.                         meetrequirements.append(stock)
163.                         meetrequirements_stock_name.append(stock_name)
164.                         PBR_list.append(pbr)
165.                         PE_list.append(PE)
166.                         DE_list.append(DE)
167.                         PEG_list.append(PEG5)
168.
169.             # Error handling
170.             except Exception as e:
171.                 error_list.append(e) #write a list with all error messages
172.
173.         # Loop for each stock in the index list
174.         for eachStock in usedindex:
175.             yahooKeystats(eachStock)
176.             time.sleep(.1)
177.
178.
179.         # print list of stocks that meet the requirements set
180.         print("The tickers for the filtered stocks are: ")
181.         print(meetrequirements)
182.
183.
184.         # Create dataframe with all companies that meet requirements with all ratios
185.         MeetRequirements_Frame = pd.DataFrame({
186.                     'Company': meetrequirements_stock_name,
187.                     'Price to Book': PBR_list,
188.                     'Price to Earnings': PE_list,
189.                     'Total Debt to Equity': DE_list,
190.                     'PEG Ratio': PEG_list,})
191.
192.
193.         # Ask user whether to print an excel file of the stocks that meet the requiremen
    ts
194.         while True:
195.             try:
196.                 requirements_excel = input(("Do you want to have an Excel file of all st
    ocks that meet the requirements including ratios? Insert Yes or No: "))
197.                 if requirements_excel.lower() == "no":
198.                     break # nothing happens
199.
200.                 elif requirements_excel.lower() == "yes":
201.                     # write excel file
202.                     with pd.ExcelWriter('MeetRequirements.xlsx') as writer:
203.                         MeetRequirements_Frame.to_excel(writer, sheet_name='Stocks with
    Ratios')
```

```
204.                      Statistics_Frame.to_excel(writer, sheet_name='Given Requirements
     ')
205.                        # give user information
206.                        print("\nWe have now saved the file named 'MeetRequirements.xlsx' in
     your current working directory.")
207.                        break # everthing ends
208.
209.               else:
210.                        print("\nError. Please insert 'Yes' or 'No'. Please try again.") # p
     rint error message
211.                        continue  # ask user again to give an input again after error messag
     e
212.
213.             # error handling for any other error
214.             except:
215.                 print("\nError. Please insert 'Yes' or 'No'. Please try again.") # print
     error message
216.                 continue  # ask user again to give an input again after error message
```

Step 3:

In the third step the user decides on whether to progress with a peer analysis or to exit. Peers are selected manually or recommended by the program. By entering 'help' in the manual setting the user gets an overview of all available peer companies with their tickers and can therefore easily find the ticker for the desired peer company. The recommendation is based on data from NASDAQ and scraped with the package 'Selenium'. 'Beautiful Soup' in this case was not able to scrape the data as the website is not html coded. Selenium also is extremely easy to use by coping xpaths in to the function *driver.find_element_by_xpath()* to target data on the respective website.

```
1.  ##############################
2.  # STEP 3: PEER GROUP ANALYSIS
3.  ##############################
4.
5.  '''''
6.  Here you get the chance to conduct a peer group analysis with some stock. The peers
7.  can be either manually inserted or you get a recommendation of peers.
8.  '''
9.
10. # Ask user whether to conduct a peer group analysis with the stock the user inserted:
11. print("\nDo you want to continue with a peer-group analysis for a chosen stock? ")
12.
13. # Check for user input and conduct the analysis if requested
14. while True:
15.     try:
16.         group_an = input(("Please type Yes or No: "))
17.         if group_an.lower() == "no":
18.             # Give user information:
19.             print("\nNo peer group analysis provided.")
20.             break #no fundamental stock analysis ends
21.
22.         elif group_an.lower() == "yes":
23.
24.             ##########################
25.             # START PEER GROUP ANALYSIS
26.             ##########################
27.
28.             # Ask user which ticker to further analyse
```

8

```python
29.          while True:
30.              try:
31.                  # Ask for user input
32.                  target_stock = input(("Please pick a stock out of the filtered stoc
     ks from above or from the SP500 / DJI and insert its ticker: "))
33.                  # check if ticker is in SP500
34.                  if target_stock.upper() in SP500:
35.                      # continue with analysis
36.                      break
37.                  # check if ticker is in DJI
38.                  elif target_stock.upper() in DJI:
39.                      break
40.                  # Let user type in another ticker which is in index
41.                  else:
42.                      print("Error. Please insert a ticker from the given indices.")
     # print error message
43.                      continue  # ask user again to give an input again after error m
     essage
44.              # error handling
45.              except:
46.                  print("\nError. Please insert a ticker from the given indices.") #
     print error message
47.                  continue
48.
49.          # Ask user how to compare stock:
50.          print("\nDo you want to insert stocks for comparison manually or continue w
     ith recommended stocks?")
51.          while True:
52.              try:
53.                  option_no = input(("Please type 1 for manual insertion or 2 for rec
     ommendations: "))
54.
55.                  #######################################
56.                  # OPTION 1: MANUAL PEER GROUP ANALYSIS
57.                  #######################################
58.                  if option_no == "1":
59.                      while True:
60.                          try:
61.                              # Ask the user for input
62.                              #print("\nA list of all possible stocks and its tickers
     can be found here:")
63.                              #print_full(Ticker_List)
64.
65.                              peers = input(("\nIf you insert 'help' a full list with
     tickers will show up where you can choose from. Otherwise, directly insert the tickers
     here separated by commas: "))
66.                              # copy input to new variable to check if user wants hel
     p
67.
68.                              # Transform user input
69.                              # split input by comma to get a list
70.                              peers = peers.split(',')
71.                              # transform each ticker to upper letters:
72.                              peers = [i.upper() for i in peers]
73.                              # delete spaces in each entry of the list
74.                              peers = [i.strip(' ') for i in peers]
75.                              # delete duplicates from peers list
76.                              peers = list(dict.fromkeys(peers))
77.
78.                              # merge indices
79.                              mergedindices = SP500 + DJI
80.                              # delete duplicates from mergedindices list
```

9

```python
81.                                    mergedindices = list(dict.fromkeys(mergedindices))

82.
83.                                # user needs help
84.                                if peers == ['HELP']:
85.                                    # print full Ticker_List (data loaded in preamble)

86.                                    print_full(Ticker_List)
87.                                    continue

88.
89.                                # user inserted something else than help
90.                                else:
91.                                    # test if all tickers exist in list check = true, o
    therwise, check = false
92.                                    check = False
93.                                    check = any(elem in peers for elem in mergedindices
    )
94.
95.                                    # if all tickers exist in list (check = true)
96.                                    if check is True:
97.                                        # Stocks we will analyse now:
98.                                        print("\nWe now continue with the peer group an
    alysis with the following tickers:")
99.                                        print(peers)
100.                                        break # if check is true ends
101.
102.                                        # if tickers do not exist in our list (check
     = false), give user chance to inser new tickers:
103.                                    else:
104.                                        print("\nError. The tickers do not exist
     in our list. Please try again.")
105.                                        continue # with user input of peer stock
     s (option 1)
106.
107.                            # error handling
108.                            except:
109.                                print("\nError. The tickers do not exist in our
     list. Please try again. ") # error message
110.                                continue  # with user input of peer stocks (opti
     on 1)
111.
112.                        break # everything ends of option number 1

113.
114.                    ###############################
115.                    # OPTION 2: PEER RECOMMENDATION
116.                    ###############################
117.                    elif option_no == "2":
118.                        print("\nPlease wait a moment while we identify your pee
    r stocks. \nThis may take a few seconds.")
119.
120.                        # SCRAPER SETTINGS
121.                        # set options for driver
122.                        options = webdriver.ChromeOptions()
123.                        options.add_argument('--ignore-certificate-errors')
124.                        options.add_argument("--test-type")
125.                        options.binary_location = "/usr/bin/chromium"
126.
127.                        # start driver
128.                        driver = webdriver.Chrome(executable_path=executable_pat
    h)
129.
130.                        # url to be visited
```

```python
131.                            url = ('https://www.nasdaq.com/symbol/' + target_stock +
        '/stock-comparison')
132.
133.                            # get request
134.                            driver.get(url)
135.
136.                            # Wait for 5 seconds
137.                            time.sleep(5)
138.
139.                            # handle cookies
140.                            cookies_list = driver.get_cookies()
141.                            cookies_dict = []
142.                            for cookie in cookies_list:
143.                                cookies_dict.append([cookie['name'],cookie['value']]
        )
144.
145.                            # create empty lists where we add peer tickers
146.                            peers = []
147.                            peer_names = []
148.
149.                            #loop to scrape tickers
150.                            for i in range(1, 5):
151.                                # scrape element
152.                                try:
153.                                    # scrape peer ticker
154.                                    peer_ticker = driver.find_element_by_xpath('//*[
        @id="quotes_content_left_SC_Symbol'+str(i)+'"]')
155.                                    # get the attribute
156.                                    peer_ticker = peer_ticker.get_attribute('value')

157.                                    # add ticker to peers list
158.                                    peers.append(peer_ticker)
159.
160.                                    # scrape peer name
161.                                    peer_name = driver.find_element_by_xpath('//*[@i
        d="quotes_content_left_SC_CompanyPanel"]/table/tbody/tr[1]/td['+str(i+1)+']/div[2]/a')

162.                                    # add name to peer_names list
163.                                    peer_names.append(peer_name.text)
164.
165.                                # error handling if there does not exist such an ele
        ment
166.                                except NoSuchElementException:
167.                                    print("Peer number" + str(i) + "out of 5 not fou
        nd!")
168.                                    pass
169.
170.                            # quit / close river
171.                            driver.quit()
172.
173.                            # create dataframe with stocks and tickers
174.                            Peer_Recommendation_Frame = pd.DataFrame({
175.                                    'Peer Name': peer_names},
176.                                    index = peers)
177.                            # set ticker as index
178.                            Peer_Recommendation_Frame.index.name = 'Ticker'
179.
180.                            # print list of peer tickers
181.                            print("\nWe identified the following peer stocks for you
        :")
182.                            print(Peer_Recommendation_Frame)
183.
```

```
184.
185.                               # Message if no peers found:
186.                               if (len(peers) == 0):
187.                                   print("Sorry, no peers found! Enter your own peers i
      f you want to.")
188.
189.                               break # option number 2 ends
190.
191.                           # if input is neither 1 or 2, give user chance for new input
      :
192.                           else:
193.                               print("\nError. Please insert a number: 1 for manual ins
      ertion or 2 for recommendations. Please try again.") # print error message if input is
      not a number
194.                               continue  # ask user again to give an input whose size w
      ill be checked again
195.
196.                       # ERROR HANDLING
197.                       except:
198.                           print("\nError. Please insert a number: 1 for manual inserti
      on or 2 for recommendations. Please try again.") # print error message if input is not
      a number
199.                           continue  # ask user again to give an input
```

Step 4:

In the fifth step a benchmark table is created and filled with data from the selected peers. The benchmark table makes it extremely easy to get an understanding of the target's valuation compared to its peers.

```
1.  ##################################
2.  # STEP 4: PEER GROUP VALUATIONS
3.  ##################################
4.
5.  #Here you will be provided with financials of the peers from STEP 3 and the company you
     wanted to analyse. The output will be tables of bechmark valuations.
6.
7.
8.  ###########################
9.  # STEP 4.1: PEER BENCHMARK
10. ###########################
11.
12.         print("\nBelow you can find some information about the benchmark stocks: \n
     ")
13.
14.         # create empty lists to be filled by loop later
15.         Forward_PE_list = []
16.         EV_Revenue_list = []
17.         EV_EBITDA_list = []
18.
19.         # Loop to scrape key statistics for chosen stocks
20.         for targetstock in peers:
21.             # url to be visited for each stock
22.             url = "https://finance.yahoo.com/quote/" + targetstock + "/key-
     statistics"
23.             # send get request for url
24.             r=requests.get(url)
25.             # parse text
26.             soup = bs.BeautifulSoup(r.text,"html.parser")
27.
```

```python
28.              # SCRAPE KEY STATISTICS
29.              # Forward / PE
30.              Forward_PE = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[3].t
     ext
31.              # append element to list
32.              Forward_PE_list.append(Forward_PE)
33.
34.              # EV / Revenue
35.              EV_Revenue = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[7].t
     ext
36.              # append element to list
37.              EV_Revenue_list.append(EV_Revenue)
38.
39.              # EV / EBITDA
40.              EV_EBITDA = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[8].te
     xt
41.              # append element to list
42.              EV_EBITDA_list.append(EV_EBITDA)
43.
44.          # Create dataframe out of lists
45.          benchmarks = pd.DataFrame({
46.                      'Forward PE': Forward_PE_list,
47.                      'EV / Revenue': EV_Revenue_list,
48.                      'EV / EBITDA': EV_EBITDA_list},
49.                      index = peers)
50.
51.          # give index name
52.          benchmarks.index.name = 'Company Ticker'
53.
54.          # Change 'N/A' to nan by pandas
55.          benchmarks = benchmarks.replace('N/A', np.nan)
56.
57.          # create list of all statistics
58.          statistics = ['Forward PE', 'EV / Revenue', 'EV / EBITDA']
59.
60.          # loop over each statistics:
61.          for stat in statistics:
62.              # change datatype for each statistics
63.              benchmarks[stat] = pd.to_numeric(benchmarks[stat])
64.
65.          # Compute mean and median and add it to the end of the dataframe
66.          benchmarks.loc['Average'] = benchmarks[:(len(benchmarks))].mean()
67.          benchmarks.loc['Median'] = benchmarks[:(len(benchmarks)-1)].median()
68.
69.          # Show user benchmark analysis
70.          print_full(benchmarks)
71.
72.
73.
74. #####################################
75. # STEP 4.2: TARGET COMPANY FINANCIALS
76. #####################################
77.
78.          # create a list of index
79.          Target_Stock = []
80.          Target_Stock.append(target_stock.upper())
81.
82.          # Create dataframe out of lists
83.          r=requests.get("https://finance.yahoo.com/quote/" + target_stock + "/key-
     statistics?p=" + target_stock)
84.          # parse text
85.          soup = bs.BeautifulSoup(r.text,"xml")
```

13

```
86.             # scrape PE Ratio
87.             Forward_PE_target = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[3
    ].text
88.             # scrape EV-Revenue
89.             EV_Revenue_target = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[7
    ].text
90.             # Scrape EV-EBITDA
91.             EV_EBITDA_target = soup.find_all('td',{'class':'Fz(s) Fw(500) Ta(end)'})[8]
    .text
92.
93.             # Create dataframe
94.             Target_Frame = pd.DataFrame({
95.                         'Forward PE': Forward_PE_target,
96.                         'EV / Revenue': EV_Revenue_target,
97.                         'EV / EBITDA': EV_EBITDA_target},
98.                         index = Target_Stock)
99.
100.                Target_Frame.index.name = 'Company Ticker'
101.
102.                Target_Frame = Target_Frame.replace('N/A', np.nan)
103.
104.                #Empty row for better overview
105.                print("\n", Target_Frame)
```

Step 5:

In the fifth step the user can select an industry benchmark against which the target stock and the peers are graphically compared. Afterwards the user can select to have an absolute price chart including a 50 day and 100 day moving average and a historical volatility plot and a rolling historical volatility plot. For the stock data we use *pdr.Datareader* which can efficiently download stock price data.

```
1.  ################
2.  # STEP 5: PLOTS
3.  ################
4.
5.  #
6.  #Here you will be provided with some plots of stock returns.
7.  #
8.
9.  ###############################
10. # STEP 5.1: INDEXED STOCK PLOT
11. ###############################
12.
13.             start = dt.datetime(2015,1,1)
14.             end = dt.datetime.today()
15.
16.             print("\nFor a good comparison of stock price development, we will show an
    indexed stock plot.\nDo you want to add an industry index for that? ")
17.
18.             # Check for user input
19.             while True:
20.                 try:
21.                     # Ask user for answer
22.                     industry_index = input(("Please type Yes or No: "))
23.
24.                     # if user doesn't want to add an industry index
25.                     if industry_index.lower() == "no":
26.                         print("The plot will be provided without an industry index.")
27.                         break # no index ends
28.
29.                     # if user wants to add an industry index
```

14

```python
                    elif industry_index.lower() == "yes":

                        # define industry indices
                        tech_index = 'IYW'
                        industrial_index = 'IYJ'
                        pharma_index = 'IHE'
                        finance_index='IYF'
                        energy_index='IYE'
                        telcom_index='IYZ'

                        # let user select industry index:
                        while True:
                            try:
                                add_index = input(("Please select your industry index.
    \nType 1 for Tech, 2 for Industrials, 3 for Pharma, 4 for Financials, 5 for Energy, 6 f
    or Telecommunications. "))
                                if add_index == "1":
                                    peers.append(tech_index)
                                    print('\nYour selected tech index is: iShares Dow J
    ones US Technology')
                                    print('\nThe Ticker of your selected Index is: ' +
    tech_index)
                                    break
                                elif add_index == "2":
                                    peers.append(industrial_index)
                                    print('\nYour selected industrial index is: iShares
     Dow Jones US Industrial')
                                    print('\nThe Ticker of your selected Index is: ' +
    industrial_index)
                                    break
                                elif add_index == "3":
                                    peers.append(pharma_index)
                                    print('\nYour selected pharma index is: iShares US
    Pharmaceuticals')
                                    print('\nThe Ticker of your selected Index is: ' +
    pharma_index)
                                    break
                                elif add_index == "4":
                                    peers.append(finance_index)
                                    print('\nYour selected finance index is: iShares Do
    w Jones US Financial Sector')
                                    print('\nThe Ticker of your selected Index is: ' +
    finance_index)
                                    break
                                elif add_index == "5":
                                    peers.append(energy_index)
                                    print('\nYour selected energy index is: iShares Dow
     Jones US Financials')
                                    print('\nThe Ticker of your selected Index is: ' +
    energy_index)
                                    break
                                elif add_index == "6":
                                    peers.append(telcom_index)
                                    print('\nYour selected telecom index is: iShares Do
    w Jones US Telecom')
                                    print('\nThe Ticker of your selected Index is: ' +
    telcom_index)
                                    break
                                else:
                                    print("\nError. Please insert an integer between 1
    and 6. Please try again.") # print error message if input is not a number
                                    continue  # ask user again to give an input
```

```python
77.
78.                            # error handling for industry index selection
79.                            except:
80.                                print("\nError. Please insert an integer between 1 and
       6. Please try again.") # print error message if input is not a number
81.                                continue  # ask user again to give an input
82.
83.                        #Target Stock is added to the peer array in order to be display
       ed in the graph
84.                        peers.append(target_stock)
85.                        break # index choice ends
86.
87.                    # if input is neither yes nor no:
88.                    else:
89.                        print("\nError. Please insert 'yes' or 'no'. Please try again."
       ) # print error message if input is not a number
90.                        continue  # ask user again to give an input
91.
92.                # error handling
93.                except:
94.                    print("\nError. Please insert 'yes' or 'no'. Please try again.") #
       print error message if input is not a number
95.                    continue  # ask user again to give an input
96.
97.            # GET DATA AND PLOT IT
98.            # function to get data from yahoo api
99.            def yahoodata(ticker_list):
100.                    # Get data from yahoo
101.                    try:
102.                        series = pdr.DataReader(ticker_list, 'yahoo', start, end)
103.                        return series
104.                    # Error handling
105.                    except Exception as e:
106.                        print('Sorry, data is currently not available: ' + str(e))
107.
108.                # get data
109.                df = yahoodata(peers)
110.
111.                # use adj close
112.                adj_price = df['Adj Close']
113.
114.                # compute stock returns
115.                stock_return = adj_price.apply(lambda x: x / x[0])
116.                stock_return.head() - 1
117.
118.                # Figure Settings
119.                # change figure size
120.                plt.rcParams['figure.figsize'] = [13.5, 9]
121.                # axis, grids and lines:
122.                stock_return.plot(grid = True).axhline(y = 1, color = "black", lw =
       1)
123.                # title
124.                plt.title("Indexed stock price performance comparison", fontsize=14,
       fontweight='bold')
125.                # label of x-axis
126.                plt.xlabel("time")
127.                # label of y-axis
128.                plt.ylabel("Indexed stock prices")
129.                # show plot
130.                plt.show()
131.
132.
```

```python
133.        ###############################################
134.        # STEP 5.2: TARGET STOCK & VOLA PLOTS
135.        ###############################################
136.
137.                    # Ask user whether he likes additional technical stock data plot
138.                    print("\nDo you want to get additional technical stock data?")
139.                    while True:
140.                        try:
141.                            # let user answer
142.                            tech_stock_data = input("Please write Yes or No: ")
143.                            # if user wants additional technical stock data:
144.                            if tech_stock_data.lower() == "yes":
145.                                # get data from yahoo
146.                                stock_data = yahoodata(target_stock)
147.                                stock_data_close = pd.DataFrame(stock_data.Close)
148.
149.                                # compute means
150.                                stock_data_close['MA_50'] = stock_data_close.Close.rolli
     ng(50).mean()
151.                                stock_data_close['MA_100'] = stock_data_close.Close.roll
     ing(100).mean()
152.
153.                                # Figure settings
154.                                # figure size
155.                                plt.figure(figsize = (15,10))
156.                                # activate grid
157.                                plt.grid(True)
158.                                # title
159.                                plt.title("Stock price with 100 day MA and 50 day MA", f
     ontsize=14, fontweight='bold')
160.                                # Plot series with respective names
161.                                plt.plot(stock_data_close['Close'], label= 'Closing Pric
     e')
162.                                plt.plot(stock_data_close['MA_100'], label= 'MA 100 day'
     )
163.                                plt.plot(stock_data_close['MA_50'], label= 'MA 50 day')

164.                                plt.legend(loc=2)
165.                                # label x-axis
166.                                plt.xlabel("time")
167.                                # label y-axis
168.                                plt.ylabel("stock price")
169.                                # show plot
170.                                plt.show()
171.
172.                                #VOLATILITY
173.                                # 1. Historic Volatility
174.                                stock_data_close['change'] = np.log(stock_data_close['Cl
     ose'] / stock_data_close['Close'].shift())
175.
176.                                # Figure settings
177.                                # figure size
178.                                plt.figure(figsize = (10,5))
179.                                # title
180.                                plt.title("Historical volatility", fontsize=14, fontweig
     ht='bold')
181.                                # Plot series
182.                                plt.plot(stock_data_close.change, color ='g')
183.                                # label x-axis
184.                                plt.xlabel("time")
185.                                # label y-axis
186.                                plt.ylabel("change")
```

```python
187.                              # show plot
188.                              plt.show()
189.
190.                              # 2. Rolling historically volatility
191.                              stock_data_close['volatility'] = stock_data_close.change
     .rolling(21).std().shift()
192.
193.                              # Figure settings
194.                              # figure size
195.                              plt.figure(figsize = (10,5))
196.                              # title
197.                              plt.title("Rolling historical volatility",fontsize=14, f
     ontweight='bold')
198.                              # plot series
199.                              plt.plot(stock_data_close.volatility, color ='orange')
200.                              # label x-axis
201.                              plt.xlabel("time")
202.                              # label y-axis
203.                              plt.ylabel("change")
204.                              # show plot
205.                              plt.show()
206.                              break # additional technical data ends
207.
208.                        # if user does not want an additional analysis:
209.                        elif tech_stock_data.lower() == "no":
210.                            print("\nNo additional technical data will be provided."
     )
211.                            break # no additional analysis ends
212.
213.
214.                        else:
215.                            print("Error. You have to enter Yes or No. Please try aga
     in.") # print error message if input is not possible
216.                            continue # ask user again to give an input
217.
218.                    except ValueError: # error handling for technical analysis
219.                        print("Error. Data currently not available. Please try again
     later. ") # print error message if input is not possible
220.                        continue # ask user again to give an input
221.
222.                    break # fundamental stock analysis ends (yes case)
223.
224.             # if continuation input for peer-
     group analysis is neither yes or no, give user another chance:
225.             else:
226.                 print("\nError. Please insert 'yes' or 'no'. Please try again.") # p
     rint error message
227.                 continue  # ask user again to give an input
228.
229.         # error handling for peer-group analysis
230.         except ValueError:
231.             print("\nError. Please insert 'yes' or 'no'. Please try again.") # print
     error message
232.             continue  # ask user again to give an input
233.
234.         break #everything ends
235.
236.     print("\nThanks for using the tool. The program will now end.")
```

3 Example

Welcome to the Fundamental Stock Screener!
This tool helps you to efficiently filter through stocks for your fundamental stock analysis and stock picking!

You can choose between two stock indices: Dow Jones Industrial (DJI) and S&P500. Which one do you want to choose?
Please enter either DJI or SP500 here: DJI

The stock index DJI will be analysed now. You now have to set the minimum requirements a stock must meet.

What is the maximum Price to Book Ratio a stock may have? Note that investors usually set a number around 2 here.
Please insert the maximum number of the Price to Book Ratio here: 2

What is the maximum Price to Earnings Ratio a stock may have? Note that investors usually set here a number around 20.
Please insert the maximum number of the Price to Earnings Ratio here: 20

What is the maximum Debt to Equity Ratio a stock may have? Set it around 100 to get a result.
Please insert the maximum number of the Debt to Equity Ratio here: 100

Great! You will now get the stocks that meet the given requirements:

	Ratio Name	Ratio Value
0	Maximum Price to Book Ratio	2.0
1	Maximum Price to Earnings Ratio	20.0
2	Maximum Debt to Equity Ratio	100.0

This process may take a few minutes. Please be patient.
CVX - Chevron Corporation meets the requirements:

Price to Book Ratio: 1.46
Price to Earnings Ratio: 16.41
Total Debt to Equity: 23.84
PEG Ratio: 0.60

XOM - Exxon Mobil Corporation meets the requirements:

Price to Book Ratio: 1.64
Price to Earnings Ratio: 17.07
Total Debt to Equity: 22.46
PEG Ratio: 1.40

TRV - The Travelers Companies, Inc. meets the requirements:

Price to Book Ratio: 1.59

Price to Earnings Ratio: 15.03

Total Debt to Equity: 30.57

PEG Ratio: 0.88

WBA - Walgreens Boots Alliance, Inc. meets the requirements:

Price to Book Ratio: 1.91

Price to Earnings Ratio: 9.72

Total Debt to Equity: 71.05

PEG Ratio: 1.71

The tickers for the filtered stocks are:

['CVX', 'XOM', 'TRV', 'WBA']

Do you want to have an Excel file of all stocks that meet the requirements including ratios? Insert Yes or No: No

Do you want to continue with a peer-group analysis for a chosen stock?

Please type Yes or No: Yes

Please pick a stock out of the filtered stocks from above or from the SP500 / DJI and insert its ticker: XOM

Do you want to insert stocks for comparison manually or continue with recommended stocks?

Please type 1 for manual insertion or 2 for recommendations: 2

We identified the following peer stocks for you:

Ticker	Peer Name
CVX	Chevron Corporation
COP	ConocoPhillips
SU	Suncor Energy Inc.
BP	BP p.l.c.

Below you can find some information about the benchmark stocks:

Company Ticker	Forward PE	EV / Revenue	EV / EBITDA
CVX	13.3200	1.65	7.810000
COP	12.8500	2.09	5.010000
SU	15.5500	NaN	NaN
BP	10.2900	0.67	5.910000
Average	13.0025	1.47	6.243333
Median	13.0850	1.65	5.910000

Company Ticker	Forward PE	EV / Revenue	EV / EBITDA
XOM	13.55	1.34	9.81

For a good comparison of stock price development, we will show an indexed stock plot.
Do you want to add an industry index for that?

Please select your industry index.
Type 1 for Tech, 2 for Industrials, 3 for Pharma, 4 for Financials, 5 for Energy, 6 for Telecommunications. 5

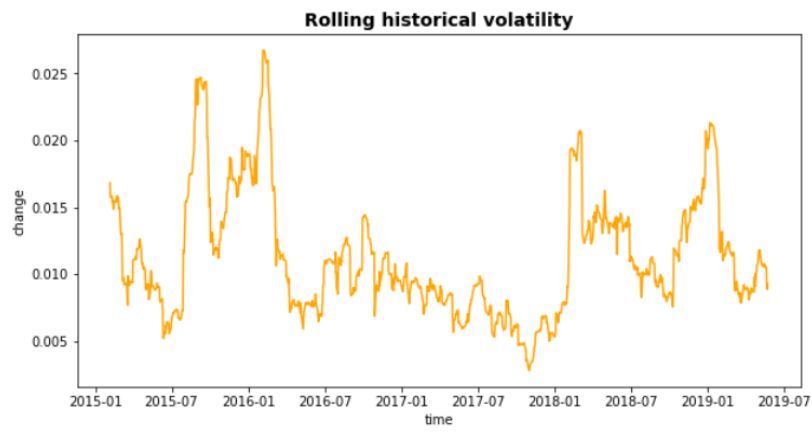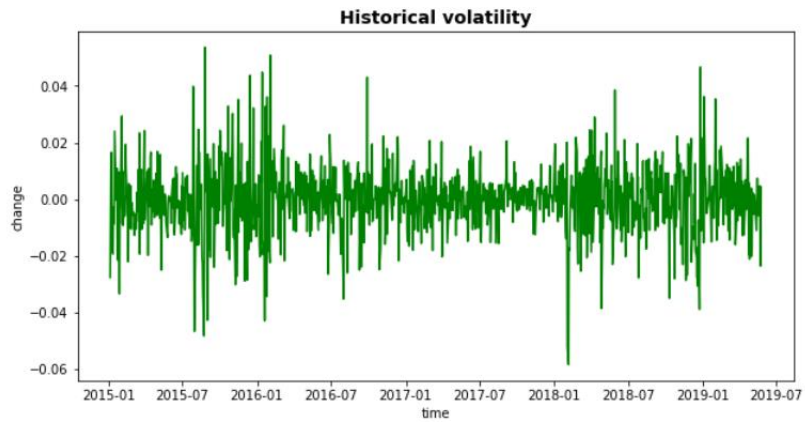Your selected energy index is: iShares Dow Jones US Financials
The Ticker of your selected Index is: IYE
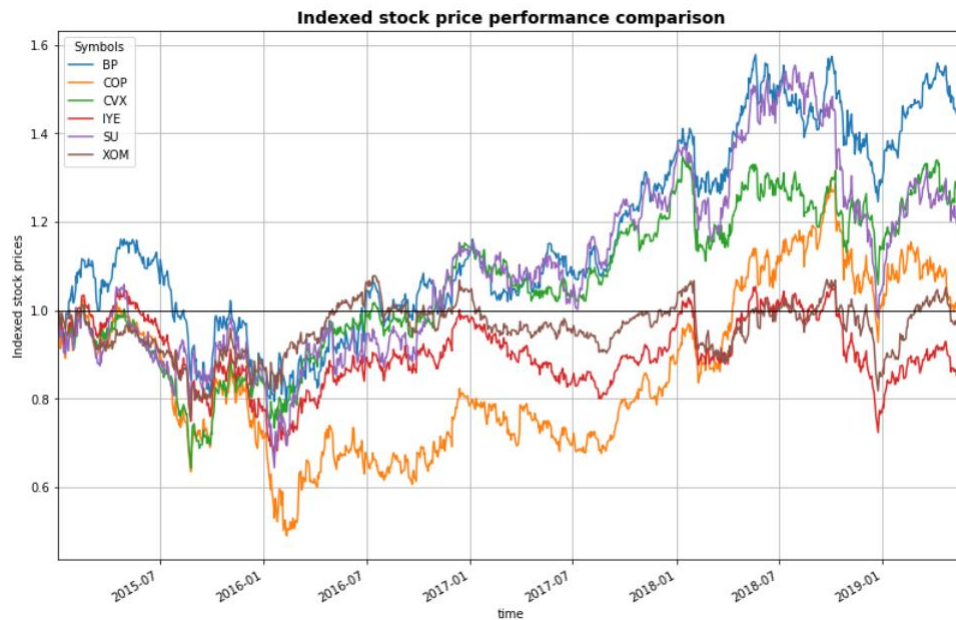
Do you want to get additional technical stock data?
Please write Yes or No: Yes

Historical volatility

Rolling historical volatility

Thanks for using the tool. The program will now end.