



Universität St.Gallen

Universität St. Gallen – Hochschule für Wirtschafts-, Rechts- und
Sozialwissenschaften (HSG)

SMI Stock Analysis

Skills: Programming with Advanced Computer Languages at University
of St. Gallen

Dr. Mario Silic

Python

December 21, 2019

Authors:

Carla Greter: 15-608-516
Colette Koch: 15-604-648

Table of content

Task and outline of the program	3
Code.....	4
Example: input and output	10

Task and outline of the program

This program lets users choose a Swiss Market Index (SMI) company to analyze the performance over an ex post time period of choice and illustrate it in two plots and a table consisting of stocks from Yahoo Finance. The SMI stock is made up of 20 of the largest and most liquid Swiss Performance Index (SPI) large- and mid-cap stocks. Investors use it as the benchmark of the overall market, to which all other investments are compared. Generally speaking, the SMI companies are attractive for investors as they are comparatively safe.

First, users can select, what stock they want to be visualized in a plot. The plot shows the 20 days rolling mean, the 200 days rolling mean and the adjusted closing price over time. Rolling means (short and long term) are used in technical analysis as an indicator that helps to smooth the price action by filtering the «noise» from random short-term price fluctuations. The adjusted closing price amends the closing price of a share to accurately reflect the value of the share after taking into account capital measures. It is considered the true price of that share.

After seeing the first plot, users have the option to change the selected SMI company or go on to see further information.

Secondly, further information such as a day's open, high and low stock price, the volume and the adjusted close over a chosen amount of days ending today is illustrated in a table. This information help investors to see increasing or decreasing dynamics. If the opening and closing are far apart, it shows strong dynamics, and if the opening and closing are close together, it shows indecision or weak dynamics. The high and low show the entire price range of the period, which is helpful in assessing volatility.

Finally, a chart showing the company in perspective of risk and expected return is shown to users to help them making a well-founded buying decision.

Code

```
#before starting the code we entered the following:
#pip install matplotlib
#pip install --user --upgrade matplotlib
#pip install pandas
#pip install pandas-datareader
#pip install numpy
#pip install datetime
from pandas_datareader import data
from pandas_datareader._utils import RemoteDataError
#for graphs
import matplotlib.pyplot as plt
from matplotlib import style
import pandas as pd
import pandas_datareader.data as web
import numpy as np
from datetime import datetime as dt
from IPython.display import display

#first, we define the functions needed

#we will devide the code in four functions

#This definition get the statistics and returns a dictionary of the
selected data
def get_stats(stock_data):
    return {
        'last': np.mean(stock_data.tail(1)),
        'short_mean': np.mean(stock_data.tail(20)),
        'long_mean': np.mean(stock_data.tail(200)),
        'short_rolling': stock_data.rolling(window=20).mean(),
        'long_rolling': stock_data.rolling(window=200).mean()
    }

#This definition cleans all the data that are not numbers in certain
columns
def clean_data(stock_data, col):
    #We use pandas to get a daterange (only weekdays)
    weekdays = pd.date_range(start=START_DATE, end=END_DATE)
    #cleans the data
    clean_data = stock_data[col].reindex(weekdays)
    #Propagates last valid observation forward, is used to fill the missing
values in the dataframe
    return clean_data.fillna(method='ffill')

#This definition creates a plot
def create_plot(stock_data, ticker):
    stats = get_stats(stock_data)
    plt.subplots(figsize=(12,8))
    plt.plot(stock_data, label=ticker)
    plt.plot(stats['short_rolling'], label='20 day rolling mean')
    plt.plot(stats['long_rolling'], label='200 day rolling mean')
    #Following labes are added to the plot to vizualize the meaning of
each line
    plt.xlabel('Date')
    plt.ylabel('Adj Close price')
    plt.legend()
    plt.title('Stock Price over Time')
```

```

plt.show()
#nicer style
style.use('ggplot')

#This definition gathers all the data of the selected SMI ticker, starting
from the entered start date and ending today
def get_data(ticker):
    try:
        #We get back the open, the close, the high and the low, the volume
        and the adjusted close from the selected stock
        stock_data = data.DataReader(ticker,
                                      'yahoo',
                                      START_DATE,
                                      END_DATE)

        #here, the date is cleaned using the clean_data definition
        adj_close = clean_data(stock_data, 'Adj Close')
        #here, the plot is created using the create_plot definition
        create_plot(adj_close, ticker)

    #expection if no data was found
    except RemoteDataError:
        print('No data found for {t}'.format(t=ticker))

# Additionally, this definition will show the high, low, open, close,
volume and adjusted close for a second selected SMI stock
def STOCK_INFOS():
    try:
        df = web.DataReader(response2, 'yahoo', START_DATE, END_DATE)
        pd.set_option('display.max_columns', 6)
        pd.set_option('display.max_rows', None)
        display(df.tail(n_columns))

    except RemoteDataError:
        print('No data found.')

def BUY_SELL():
    try:
        plt.scatter(retscomp.mean(), retscomp.std())
        plt.xlabel('Expected returns')
        plt.ylabel('Risk')
        for label, x, y in zip(retscomp.columns, retscomp.mean(),
                               retscomp.std()):
            plt.annotate(
                label,
                xy = (x, y), xytext = (20, -20),
                textcoords = 'offset points', ha = 'right', va = 'bottom',
                bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow',
                               alpha = 0.5),
                arrowprops = dict(arrowstyle = '->', connectionstyle =
'arc3,rad=0'))
        plt.show()

    except RemoteDataError:
        print('No data found for')

```

#step 1: onboarding

```
print("This program lets you choose a SMI stock from Yahoo Finance to  
evaluate the performance over an chosen time period. \nFirst, we will show  
a plot that illustrates the 20 days rolling mean, the 200 days rolling mean  
and the stock price. \nSecond, a table will show additional information  
about a selected SMI stock for the chosen period. \nEventually, a chart  
showing the risk and expected returns will be illustrated.")
```

```
#The end date of the stock analysis is always today  
END_DATE = str(dt.now().strftime('%Y-%m-%d'))
```

```
run_again = ("yes")  
while run_again == ("yes"):
```

```
    while True:  
        #here, we let the user pick a date where the stock analysis  
        should start  
        START_DATE = str(input('What start date do you want? Please enter a  
date in the past in the following format YYYY-MM-DD: '))
```

```
        try:  
            dt.strptime(START_DATE, '%Y-%m-%d')  
            print('The date {} is valid.'.format(START_DATE))  
            break  
        except ValueError:  
            print('The date {} is invalid'.format(START_DATE))  
            continue
```

```
    run_again1 = ("no")  
    while run_again1 == ("no"):
```

```
        #pre-entered SMI stocks including the abbreviation to fill in  
        get_data  
        while True:  
            # The comparison works regardless of capital or lower letters  
            stock_data = str.upper(input("What SMI stock are you  
interested in? "))
```

```
            if stock_data == 'NESTLE':  
                get_data('NESN.SW')  
                break  
            elif stock_data == 'ROCHE':  
                get_data('ROG.SW')  
                break  
            elif stock_data == 'NOVARTIS':  
                get_data('NOVN.SW')  
                break  
            elif stock_data == 'UBS':  
                get_data('UBSG.SW')  
                break  
            elif stock_data == 'ABB':  
                get_data('ABBN.SW')  
                break  
            elif stock_data == 'RICHEMONT':  
                get_data('CFR.SW')  
                break
```

```

elif stock_data == 'ZURICH':
    get_data('ZURN.SW')
    break
elif stock_data == 'CREDIT SUISSE':
    get_data('CSGN.SW')
    break
elif stock_data == 'SWISSRE':
    get_data('SREN.SW')
    break
elif stock_data == 'LAFARGE HOLCIM':
    get_data('LHN.SW')
    break
elif stock_data == 'LONZA':
    get_data('LONN.SW')
    break
elif stock_data == 'GIVAUDAN':
    get_data('GIVN.SW')
    break
elif stock_data == 'SIKA':
    get_data('SIKA.SW')
    break
elif stock_data == 'GEBERIT':
    get_data('GEBN.SW')
    break
elif stock_data == 'SGS':
    get_data('SGSN.SW')
    break
elif stock_data == 'ALCON':
    get_data('ALC.SW')
    break
elif stock_data == 'SWISSCOM':
    get_data('SCMN.SW')
    break
elif stock_data == 'ADECCO':
    get_data('ADEN.SW')
    break
elif stock_data == 'SWATCH':
    get_data('UHR.SW')
    break
elif stock_data == 'SWISSLIFE':
    get_data('SLHN.SW')
    break

else:
    #if users enteres a non-SMI stock / wrong spelling, they
    can try again
    print("Please enter a valid SMI stock. Try Again: ")
    continue

    run_again1 = input("Do you want to see additional information for
    this stock? If 'no', you can choose another one \n(yes/no) ")

    if run_again1 == ("no"):
        print ("Alright, let's choose another stock!")
        continue

    elif run_again1 == ("yes"):

```

```

        print ("The table will show you the open, high and low stock
price,\nthe volume and the adjusted close over a chosen amount of days
ending today.")

```

```

while True:
    #Users can choose howw many days they want to see
    n_columns = int(input("How many days do you want to be showed? "))

    if stock_data == 'NESTLE':
        response2 = 'NESN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'ROCHE':
        response2 = 'ROG.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'NOVARTIS':
        response2 = 'NOVN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'UBS':
        response2 = 'UBSG.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'ABB':
        response2 = 'ABBN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'ZURICH':
        response2 = 'ZURN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'CREDITSUISSE':
        response2 = 'CSGN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'SWISSRE':
        response2 = 'SREN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'LAFARGEHOLCIM':
        response2 = 'LHN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'LONZA':
        response2 = 'LONN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'GIVAUDAN':
        response2 = 'GIVN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'SIKA':
        response2 = 'SIKA.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'GEBERIT':
        response2 = 'GEBN.SW'

```



```

        STOCK_INFOS()
        break
    elif stock_data == 'SGS':
        response2 = 'SGSN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'ALCON':
        response2 = 'ALC.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'SWISSCOM':
        response2 = 'SCMN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'ADECCO':
        response2 = 'ADEN.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'SWATCH':
        response2 = 'UHR.SW'
        STOCK_INFOS()
        break
    elif stock_data == 'SWISSLIFE':
        response2 = 'SLHN.SW'
        STOCK_INFOS()
        break

    print("\nLet's now look at buying opportunities. The following plot
illustrates the risk and expected return of a choosen SMI-stock. ")
    dfcomp=web.DataReader([response2],'yahoo', start = START_DATE, end =
END_DATE)['Adj Close']
    retscomp = dfcomp.pct_change()
    corr = retscomp.corr()
    BUY_SELL()

    run_again = input("Would you like to start the programm all over again?
(yes/no) ")
    if run_again == ("yes"):
        print ("Alright, let ´s start over!")

    elif run_again == ("no"):
        print ('\n0kay, hope you enjoyed our program – good bye!
\n\nKind regards \n\nColette and Carla')

```

Example: input and output

*** User Input is illustrated in blue ***

This program lets you choose a SMI stock from Yahoo Finance to evaluate the performance over a chosen time period.

First, we will show a plot that illustrates the 20 days rolling mean, the 200 days rolling mean and the stock price.

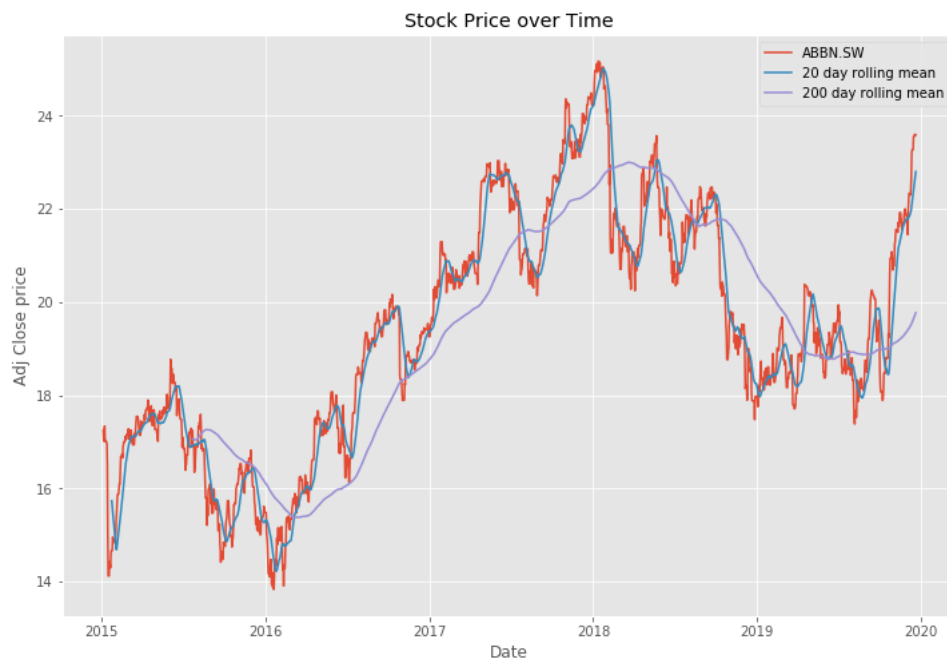
Second, a table will show additional information about a selected SMI stock for the chosen period.

Eventually, a chart showing the risk and expected returns will be illustrated.

What start date do you want? Please enter a date in the past in the following format YYYY-MM-DD: 2015-01-01

The date 2015-01-01 is valid.

What SMI stock are you interested in? abb

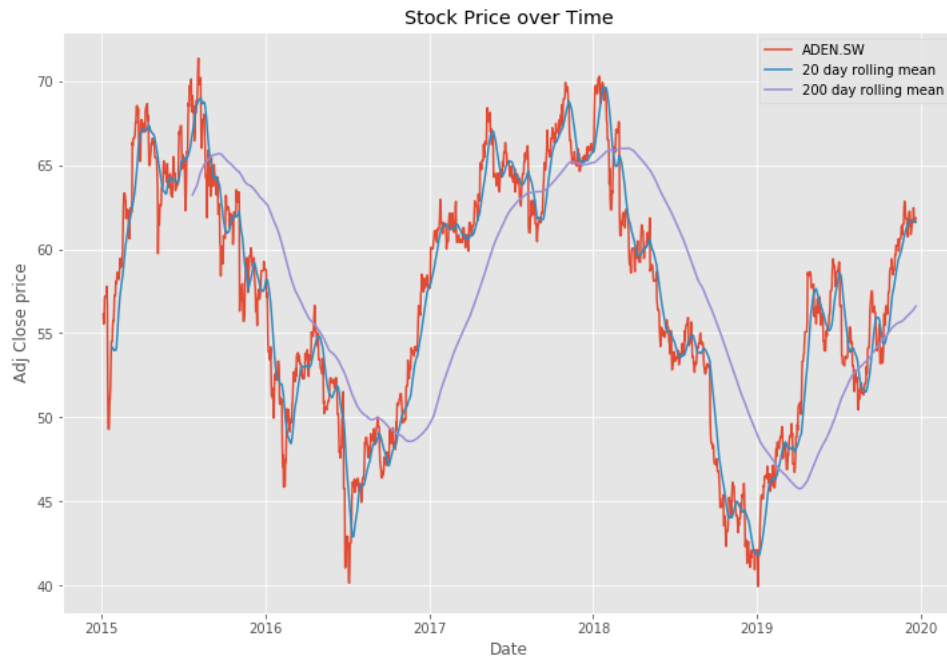


Do you want to see additional information for this stock? If 'no', you can choose another one

(yes/no) no

Alright, let's choose another stock!

What SMI stock are you interested in? adecco



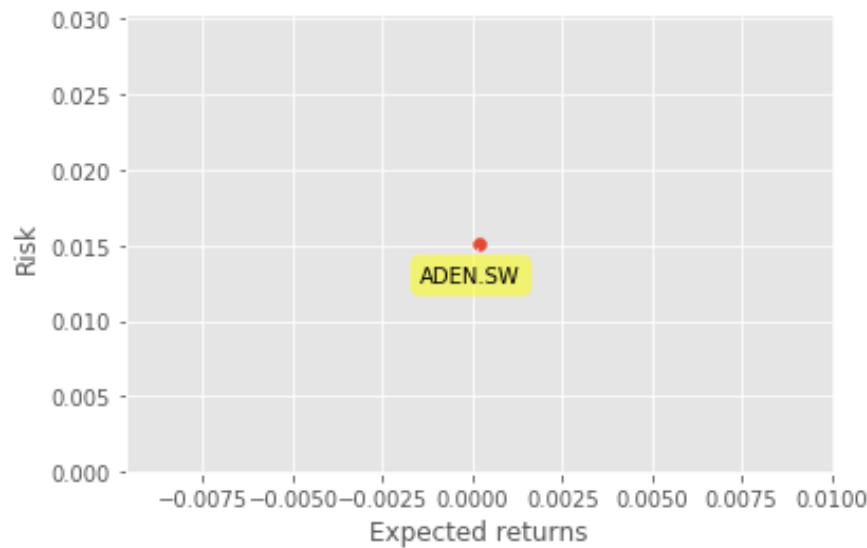
Do you want to see additional information for this stock? If 'no', you can choose another one.
(yes/no) [yes](#)

The table will show you the open, high and low stock price, the volume and the adjusted close over a chosen amount of days ending today.

How many days do you want to be showed? [8](#)

	High	Low	Open	Close	Volume	Adj
Close						
Date						
2019-12-11	61.180000	60.459999	60.799999	61.099998	770290.0	61.099998
2019-12-12	61.639999	60.680000	60.900002	61.279999	759611.0	61.279999
2019-12-13	62.820000	61.619999	61.720001	61.619999	1126352.0	61.619999
2019-12-16	62.520000	61.840000	61.840000	62.459999	787960.0	62.459999
2019-12-17	62.700001	61.900002	62.480000	61.900002	914941.0	61.900002
2019-12-18	62.400002	61.680000	61.779999	61.840000	764970.0	61.840000
2019-12-19	61.880001	61.259998	61.880001	61.580002	613707.0	61.580002
2019-12-20	62.139999	61.540001	62.139999	61.840000	1197820.0	61.840000

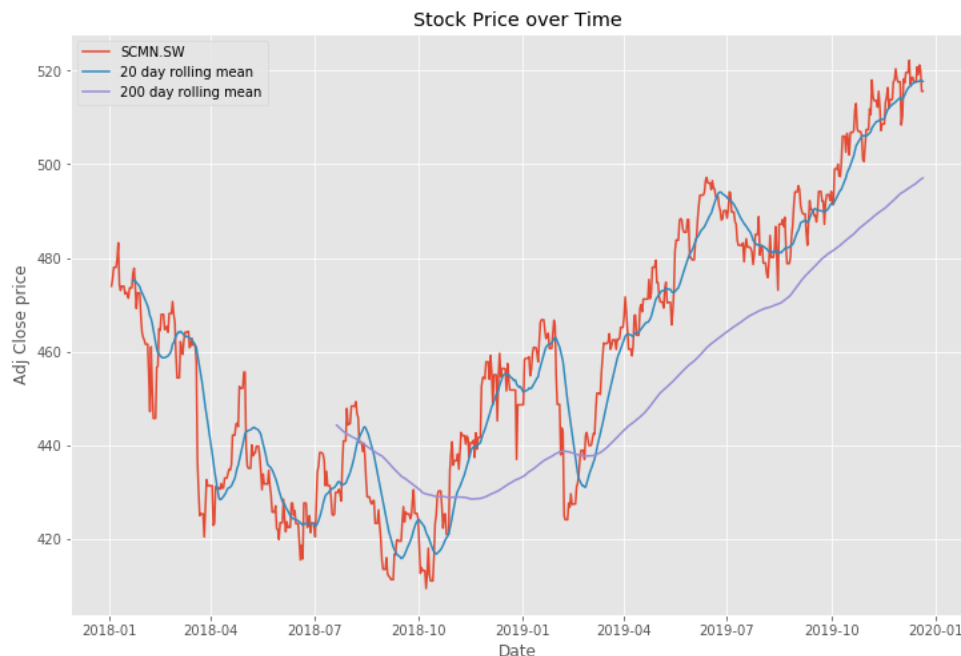
Let's now look at buying opportunities. The following plot illustrates the risk and expected return of a chosen SMI stock.



Would you like to start the program all over again? (yes/no) [yes](#)
 Alright, let's start over!

What start date do you want? Please enter a date in the past in the following format YYYY-MM-DD: [2018-01-01](#)
 The date 2018-01-01 is valid.

What SMI stock are you interested in? [swisscom](#)



Do you want to see additional information for this stock? If 'no', you can choose another one
 (yes/no) [yes](#)

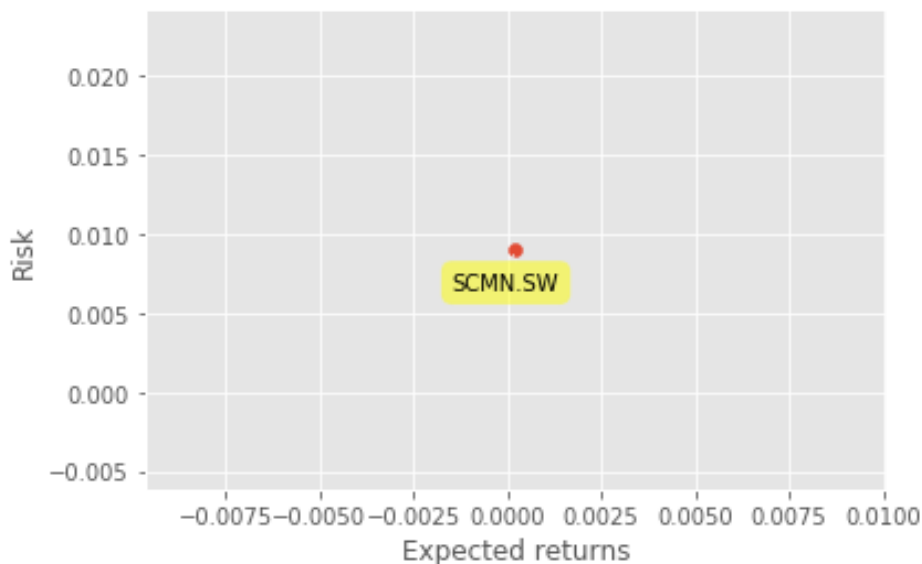
The table will show you the open, high and low stock price, the volume and the adjusted close over a chosen amount of days ending today.

How many days do you want to be showed? 5

	High	Low	Open	Close	Volume \
Date					
2019-12-16	521.400024	518.799988	518.799988	520.799988	142458.0
2019-12-17	522.599976	517.000000	522.000000	519.200012	160948.0
2019-12-18	523.400024	518.200012	519.400024	521.200012	191162.0
2019-12-19	521.799988	516.400024	521.599976	520.000000	139574.0
2019-12-20	518.000000	514.200012	518.000000	515.599976	418335.0

	Adj Close
Date	
2019-12-16	520.799988
2019-12-17	519.200012
2019-12-18	521.200012
2019-12-19	520.000000
2019-12-20	515.599976

Let's now look at buying opportunities. The following plot illustrates the risk and expected return of a chosen SMI stock.



Would you like to start the program all over again? (yes/no) no

Okay, hope you enjoyed our program – goodbye!

Kind regards

Colette and Carla