1_Data_processing

June 20, 2022

0.1 In the first part of the code, the data is processed and let ready for the computations.

0.1.1 First the data from WRDS is loaded and cleaned.

[2]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
data = pd.read_csv("data_wrds.csv",engine="python")

data.index = data["date"]
del data["date"]

data = data.drop('29/10/2012')
data = data.drop('30/09/2017')
data = data.drop('23/04/2005')

data.index = pd.to_datetime(data.index,dayfirst=True)
```

0.1.2 Return data is cleaned

[2]:
```
returns = pd.DataFrame(index=data.index,columns=["RETURNS"])

for i in range(0,len(data["RET"])):

    if data["RET"][i] == "C":

        pass

    elif data["RET"][i] == "B":

        pass

    else:
        returns.iloc[i] = float(data["RET"][i])

data["RETURNS"] = returns
```

```
[2]:             PERMNO                    COMNAM  PERMCO     CUSIP       RET  \
date
1996-01-02   10057  ACME CLEVELAND CORP NEW   20020  00462610  0.000000
1996-01-03   10057  ACME CLEVELAND CORP NEW   20020  00462610  0.020000
1996-01-04   10057  ACME CLEVELAND CORP NEW   20020  00462610 -0.026144
1996-01-05   10057  ACME CLEVELAND CORP NEW   20020  00462610  0.000000
1996-01-08   10057  ACME CLEVELAND CORP NEW   20020  00462610  0.000000
…              …                        …   …      …         …
2021-12-27   93436                TESLA INC   53453  88160R10  0.025248
2021-12-28   93436                TESLA INC   53453  88160R10 -0.005000
2021-12-29   93436                TESLA INC   53453  88160R10 -0.002095
2021-12-30   93436                TESLA INC   53453  88160R10 -0.014592
2021-12-31   93436                TESLA INC   53453  88160R10 -0.012669

                   BID         ASK     SHROUT   RETURNS
date
1996-01-02    18.75000    19.00000     6313.0       0.0
1996-01-03    18.75000    19.25000     6313.0      0.02
1996-01-04         NaN         NaN     6313.0 -0.026144
1996-01-05    18.37500    18.75000     6313.0       0.0
1996-01-08    18.37500    18.75000     6313.0       0.0
…                  …          …         …        …
2021-12-27  1093.81995  1093.83997  1004265.0  0.025248
2021-12-28  1088.68005  1089.19995  1004265.0    -0.005
2021-12-29  1085.56995  1085.88000  1004265.0 -0.002095
2021-12-30  1070.27002  1070.32996  1004265.0 -0.014592
2021-12-31  1056.89001  1057.23999  1004265.0 -0.012669

[5726276 rows x 9 columns]
```

0.1.3 A list for the name of every security is created, so every ticker is labelled for the last name its security had.

```
[3]: names = pd.DataFrame(index=data["PERMNO"].unique(),columns=["Name"])

     for i in data["PERMNO"].unique():
         names.loc[i] = data[data["PERMNO"]==i]["COMNAM"].unique()[-1]
```

0.1.4 The same is done for company names, so we only remain with companies. Also, ETFs and Trusts on the sample are removed.

```
[4]: pre_permco_list = list(data["PERMCO"].unique())

     delete_permcos =␣
      ↪[22100,29010,29548,29941,30421,32030,37493,39147,44072,45684,45874,46673,50699,50846,51187,
     46673,53120,57105]
```

```
permco_list = [permco for permco in pre_permco_list if permco not in␣
 ↪delete_permcos]

names_company = pd.DataFrame(index=permco_list,columns=["Name"])

for i in permco_list:
    names_company.loc[i] = data[data["PERMCO"]==i]["COMNAM"].unique()[-1]
```

0.1.5 A data frame for ask price, shares outstanding and returns data from every company is created

[5]:
```
ask = pd.DataFrame(index=data.index.unique(),columns=data["PERMNO"].unique())

for i in ask.columns:

   ask[i] = data[data["PERMNO"]==i]["ASK"]

ask.columns = names["Name"]

ask = ask[list(names_company["Name"])]

ask = ask.loc[:,~ask.columns.duplicated()]
```

[6]:
```
shares = pd.DataFrame(index=data.index.unique(),columns=data["PERMNO"].unique())

for i in shares.columns:

   shares[i] = data[data["PERMNO"]==i]["SHROUT"]

shares.columns = names["Name"]

shares = shares[list(names_company["Name"])]

shares = shares.loc[:,~shares.columns.duplicated()]

### million of shares outstanding

shares = shares/1000000
```

[ ]:
```
returns = pd.DataFrame(index=data.index.unique(),columns=data["PERMNO"].
 ↪unique())

for i in returns.columns:
    returns[i] = data[data["PERMNO"]==i]["RETURNS"]

returns.columns = names["Name"]

returns = returns[list(names_company["Name"])]

returns = returns.loc[:,~returns.columns.duplicated()]
```

0.1.6 Market cap is calculated and the eligility of every company is determined

[8]:
```
market_cap = ask*shares

eom_market_cap = market_cap.resample("M").pad()

eligibility = pd.DataFrame(index=eom_market_cap.index,columns=eom_market_cap.
 ↪columns)

for m in range(0,len(eom_market_cap)):

    for cap in range(0,len(eom_market_cap.iloc[m])):

        if eom_market_cap.iloc[m,cap] > 5:

            eligibility.iloc[m,cap] = 1

        else:
            eligibility.iloc[m,cap] = np.nan
```

0.1.7 Return data is filtered so it only contains companies that are elegible

[10]:
```
comp_filtered_by_cap = pd.DataFrame(eligibility.sum().
 ↪sort_values(),columns=["Months wirh marketcap > 5"])

comp_filtered_by_cap = comp_filtered_by_cap[comp_filtered_by_cap==0].dropna().
 ↪index

large_caps = [col for col in returns.columns if col not in␣
 ↪list(comp_filtered_by_cap)]

returns = returns[large_caps]
```

0.1.8 Returns are compounder to monthly frequency, and are controlled for companies that are no longer trading

[11]:
```
monthly_returns = returns.resample("M").agg(lambda x: (x + 1).prod() - 1)

active = pd.DataFrame(index=monthly_returns.index,columns=monthly_returns.
 ↪columns)

for m in range(0,len(monthly_returns)):

    for stock in range(0,len(monthly_returns.iloc[m])):

        if monthly_returns.iloc[m,stock] == 0:

            active.iloc[m,stock] = 0

        else:
            active.iloc[m,stock] = 1
```

0.1.9 A price index is constructed

[12]:
```
price_index_ = (1+monthly_returns).cumprod()

price_index = eligibility[large_caps].shift().ffill().fillna(0) * active *␣
 ↪price_index_
```

0.1.10 Data is saved

[ ]:
```
monthly_returns.to_excel("monthly_returns.xlsx")
price_index.to_excel("price_index.xlsx")
```