04_build_us_stock_dataset

September 29, 2021

1 Download historical equity data for NASDAQ stocks from yahoo finance

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
from time import time
from tqdm import tqdm
from pathlib import Path
import pandas as pd

from pandas_datareader.nasdaq_trader import get_nasdaq_symbols
import yfinance as yf
```

[ ]:
```
idx = pd.IndexSlice
```

[ ]:
```
results_path = Path('results', 'asset_pricing')
if not results_path.exists():
    results_path.mkdir(parents=True)
```

[ ]:
```
def chunks(l, n):
    for i in range(0, len(l), n):
        yield l[i:i + n]
```

[ ]:
```
def format_time(t):
    """Return a formatted time string 'HH:MM:SS
    based on a numeric time() value"""
    m, s = divmod(t, 60)
    h, m = divmod(m, 60)
    return f'{h:0>2.0f}:{m:0>2.0f}:{s:0>2.0f}'
```

1.1 Get NASDAQ symbols

[ ]:
```
traded_symbols = get_nasdaq_symbols()
```

[ ]:
```
traded_symbols.info()
```

1.2 Download metadata from yahoo finance

1.2.1 NASDAQ symbols

[ ]:
```
all_symbols = traded_symbols[~traded_symbols.ETF].index.unique().to_list()
n = len(all_symbols)
print(f'# Symbols: {n:,.0f}')
```

[ ]:
```
yf_symbols = yf.Tickers(all_symbols)
```

[ ]:
```
meta_data = []
start = time()
for ticker, yf_object in tqdm(yf_symbols.tickers.items()):
    try:
        s = pd.Series(yf_object.get_info())
        meta_data.append(s.to_frame(ticker))
    except Exception as e:
        # track errors
        print(symbol.ticker, e)

print(f'Success: {len(meta_data):5,.0f} / {n:5,.0f}')
```

[ ]:
```
df = pd.concat(meta_data, axis=1).dropna(how='all').T
df = df.apply(pd.to_numeric, errors='ignore')
df.info(show_counts=True)
```

[ ]:
```
info.to_hdf(results_path / 'data.h5', 'stocks/info')
```

1.3 Download adjusted price data using yfinance

[ ]:
```
prices_adj = []
start = time()
for i, chunk in enumerate(chunks(all_symbols, 100), 1):
    prices_adj.append(yf.download(chunk, period='max', auto_adjust=True).
 ↪stack(-1))

    per_ticker = (time()-start) / (i * 100)
    to_do = n - (i * 100)
    to_go = to_do * per_ticker
    print(f'Success: {len(prices_adj):5,.0f}/{i:5,.0f} | To go:␣
 ↪{format_time(to_go)} ({to_do:5,.0f})')
```

[ ]:
```
prices_adj = (pd.concat(prices_adj)
              .dropna(how='all', axis=1)
              .rename(columns=str.lower)
              .swaplevel())
```

[ ]:
```
prices_adj.index.names = ['ticker', 'date']
```

[ ]:
```
len(prices_adj.index.unique('ticker'))
```

1.3.1 Remove outliers

[ ]:
```
df = prices_adj.close.unstack('ticker')
pmax = df.pct_change().max()
pmin = df.pct_change().min()
to_drop = pmax[pmax > 1].index.union(pmin[pmin<-1].index)
len(to_drop)
```

[ ]:
```
prices_adj = prices_adj.drop(to_drop, level='ticker')
```

[ ]:
```
len(prices_adj.index.unique('ticker'))
```

[ ]:
```
prices_adj.sort_index().loc[idx[:, '1990': '2019'], :].to_hdf(results_path /␣
↪'data.h5',
                                                              'stocks/prices/
↪adjusted')
```