Hi,

wanted to share a simple script on how to download all available SEC 10-Q filings from IEX Cloud for all US companies.

Might be useful for you models

Can be obtained for ~$70 from IEX

Have fun!

import requests import json import pandas as pd import numerapi

In[select tickers]

napi = numerapi.SignalsAPI()

ticker_map = pd.read_csv('https://numerai-signals-public-data.s3-us-west-2.amazonaws.com/signals_ticker_map_w_bbg.csv') ticker_map.to_csv('signals_ticker_map_w_bbg.csv', index=False) print(f"Number of tickers in map: {len(ticker_map)}")

read in list of active Signals tickers which can change slightly era to era

eligible_tickers = pd.Series(napi.ticker_universe(), name='numerai_ticker') print(f"Number of eligible tickers: {len(eligible_tickers)}")

filter on US stocks only

eligible_tickers = eligible_tickers[eligible_tickers.str.contains(' US')]

map eligible numerai tickers to yahoo finance tickers

yfinance_tickers = eligible_tickers.map(dict(zip(ticker_map['bloomberg_ticker'], ticker_map['yahoo']))).dropna() numerai_tickers = ticker_map['bloomberg_ticker'] print(f'Number of eligible, mapped tickers: {len(yfinance_tickers)}')

In[Load data]

sandbox = True domain = 'https://sandbox.iexapis.com/stable' if sandbox else 'https://cloud.iexapis.com/stable' api_token = 'xxx' if sandbox else 'xxx'

session = requests.session() params = {} params["token"] = api_token headers = {"project": "sandbox"} full_data = []

for symbol in yfinance_tickers:

```
url = domain + f'/time-series/reported_financials/{symbol}/10-Q/?from=2005-01-01&format=json'
response = session.get(url=url, params=params, headers=headers)

if response.status_code == 200:

    df = pd.DataFrame( json.loads(response.text) )
    df['ticker'] = symbol
    full_data.append(df)

    print(f'RESPONSE {symbol}: {response.status_code}, cols: {len(df.columns)}')

else:
    print(f"RESPONSE {symbol}: {response.status_code}")
    print(f'error: {response}')
    break
```

In[Save last step]

print('Saving...') full_data_df = pd.concat(full_data) counts = full_data_df.isna().sum().sort_values() indexes = counts[counts<full_data_df.shape[0]*0.8].index full_data_df = full_data_df[indexes]

```
full_data_df.to_csv('financials_full_final_sandbox.csv')
```