01_explore_crsp

January 29, 2026

1 CRSP Data Exploration

Explore the structure and contents of the CRSP data files.

[26]:
```
import pandas as pd
import numpy as np
from pathlib import Path

DATA_PATH = Path("../US_CRSP_NYSE/")
```

1.1 1. Available Files

[27]:
```
# List all files
for f in DATA_PATH.rglob("*.csv"):
    print(f"{f.relative_to(DATA_PATH)} - {f.stat().st_size / 1e6:.1f} MB")
```

```
Sectors/Sectors_SP500_YahooNWikipedia.csv - 0.0 MB
Sectors/Sectors_SP1500.csv - 0.0 MB
Matrix_Format_SubsetUniverse/OPCL_20000103_20201231.csv - 33.7 MB
Matrix_Format_SubsetUniverse/pvCLCL_20000103_20201231.csv - 33.7 MB
Matrix_Format_SubsetUniverse/volMM_20000103_20201231.csv - 17.0 MB
Matrix_Format_SubsetUniverse/volume_20000103_20201231.csv - 26.4 MB
```

1.2 2. Main Data Files (Matrix Format)

Files: - `OPCL`: Open-to-Close returns - `pvCLCL`: Previous Close-to-Close returns
- `volume`: Trading volume - `volMM`: Volume in millions

[28]:
```
# Load close-to-close returns (main return series)
returns_df = pd.read_csv(DATA_PATH / "Matrix_Format_SubsetUniverse/
 ↪pvCLCL_20000103_20201231.csv", index_col=0)
print(f"Shape: {returns_df.shape}")
print(f"Tickers (rows): {returns_df.shape[0]}")
print(f"Dates (cols): {returns_df.shape[1]}")
returns_df.head()
```

```
Shape: (695, 5279)
Tickers (rows): 695
Dates (cols): 5279
```

[28]:
```
        X20000103  X20000104  X20000105  X20000106  X20000107  X20000110  \
ticker
AA      -0.024849   0.004633   0.057648  -0.013081  -0.002946  -0.002954
ABM     -0.003067   0.003077  -0.012270   0.003106   0.003096   0.000000
ABT     -0.036145  -0.028571  -0.001838   0.034991   0.010676  -0.007042
ADI     -0.030242  -0.050589   0.014599  -0.027338   0.028107   0.087050
ADM     -0.010309  -0.010417  -0.015789   0.005348   0.015957   0.000000

        X20000111  X20000112  X20000113  X20000114  …  X20201217  X20201218  \
ticker                                              …
AA      -0.005926  -0.008942  -0.018045  -0.019908  …   0.027804  -0.007665
ABM     -0.009259   0.004611  -0.031250   0.045161  …   0.006693  -0.016745
ABT     -0.014184  -0.014892  -0.009174   0.022222  …   0.013415   0.001747
ADI     -0.041032   0.032436   0.003342   0.080613  …   0.006863   0.005425
ADM     -0.015707   0.026596  -0.005181   0.041667  …   0.003659   0.006075

        X20201221  X20201222  X20201223  X20201224  X20201228  X20201229  \
ticker
AA       0.004543  -0.022614   0.028228  -0.011701   0.012750  -0.008993
ABM     -0.030804  -0.006718  -0.006243  -0.012827   0.019623  -0.018986
ABT     -0.008076   0.001758  -0.007665   0.008376  -0.005168   0.005010
ADI     -0.013628   0.011852  -0.007347   0.008728   0.001454  -0.007050
ADM     -0.006240  -0.012963   0.012723   0.001824   0.005259  -0.005030

        X20201230  X20201231
ticker
AA       0.041289   0.004357
ABM      0.000530   0.002650
ABT      0.001015   0.009683
ADI      0.017541   0.010603
ADM      0.009302   0.010018

[5 rows x 5279 columns]
```

[29]:
```python
# Check date range
dates = returns_df.columns.tolist()
print(f"First date: {dates[0]}")
print(f"Last date: {dates[-1]}")
print(f"Total trading days: {len(dates)}")
```

```
First date: X20000103
Last date: X20201231
Total trading days: 5279
```

[30]:
```python
# Sample tickers
print("Sample tickers:")
print(returns_df.index[:20].tolist())
```

```
Sample tickers:
['AA', 'ABM', 'ABT', 'ADI', 'ADM', 'ADX', 'AEE', 'AEG', 'AEM', 'AEP', 'AES',
'AFG', 'AFL', 'AIG', 'AIN', 'AIR', 'AIV', 'AJG', 'ALB', 'ALK']
```

[31]:
```
# Check for missing values
missing_pct = returns_df.isna().sum().sum() / returns_df.size * 100
print(f"Missing values: {missing_pct:.2f}%")
```

```
Missing values: 0.00%
```

[32]:
```
# Load volume data
volume_df = pd.read_csv(DATA_PATH / "Matrix_Format_SubsetUniverse/
 ↪volume_20000103_20201231.csv", index_col=0)
print(f"Volume shape: {volume_df.shape}")
volume_df.head()
```

```
Volume shape: (695, 5279)
```

[32]:

ticker	X20000103	X20000104	X20000105	X20000106	X20000107	X20000110	\
AA	1551299	2234799	3121599	4494699	4534699	3835799	
ABM	120800	62400	27400	63900	60500	113100	
ABT	4774099	4818899	5262299	7846599	7072899	4687500	
ADI	1827799	1266599	1614000	1300500	945300	1285000	
ADM	893200	986900	986800	816300	1076000	1346500	

ticker	X20000111	X20000112	X20000113	X20000114	…	X20201217	X20201218	\
					…			
AA	2231599	1873599.0	2095399.0	1878500	…	7318439	5137275	
ABM	37900	63800.0	104600.0	57000	…	826980	1003569	
ABT	4279500	3671000.0	4765500.0	5035599	…	3924302	10669778	
ADI	963100	1421199.0	842500.0	2344399	…	2351444	4221961	
ADM	991400	1231299.0	2666699.0	2544699	…	3056404	5635442	

ticker	X20201221	X20201222	X20201223	X20201224	X20201228	X20201229	\
AA	3869440	4197039	3480750	1075501	4336333	3492994	
ABM	626720	332810	324384	127465	228931	198581	
ABT	4544274	3426649	3149690	1451492	2034386	2834686	
ADI	1818703	2161211	2195585	465144	1624772	1127407	
ADM	1859094	1600118	1692926	411610	1205701	1354217	

ticker	X20201230	X20201231
AA	3435025	3443724
ABM	207225	254982
ABT	2416104	2982245
ADI	1133590	1615035

```
ADM       1533170    1698888

[5 rows x 5279 columns]
```

[33]:
```
# Load open-to-close returns
opcl_df = pd.read_csv(DATA_PATH / "Matrix_Format_SubsetUniverse/
 ↪OPCL_20000103_20201231.csv", index_col=0)
print(f"OPCL shape: {opcl_df.shape}")
opcl_df.head()
```

```
OPCL shape: (695, 5279)
```

[33]:

ticker	X20000103	X20000104	X20000105	X20000106	X20000107	X20000110	\
AA	-0.013042	0.010043	0.047628	-0.011713	-0.016118	-0.032073	
ABM	-0.009188	0.012346	-0.006192	0.000000	0.003091	0.000000	
ABT	-0.007117	-0.012786	0.011111	0.032553	0.028573	-0.021053	
ADI	-0.036071	-0.044261	0.014493	-0.027719	0.033654	0.048129	
ADM	0.000000	0.005277	-0.015915	0.010695	0.005249	-0.005222	

ticker	X20000111	X20000112	X20000113	X20000114	…	X20201217	X20201218	\
					…			
AA	0.022608	-0.005249	-0.018210	-0.020109	…	-0.004498	-0.009046	
ABM	-0.009302	0.003130	-0.028619	0.012423	…	0.045594	-0.020819	
ABT	0.010850	-0.021779	-0.009217	0.000000	…	0.008772	0.001102	
ADI	-0.031921	0.034686	-0.020443	0.057086	…	0.001601	0.001246	
ADM	-0.015831	0.020943	0.000000	0.030459	…	-0.008269	0.007476	

ticker	X20201221	X20201222	X20201223	X20201224	X20201228	X20201229	\
AA	0.039673	-0.025584	0.019083	-0.017156	0.006315	-0.004527	
ABM	-0.010283	-0.005448	-0.016100	-0.011338	0.009932	-0.024098	
ABT	0.004915	0.004165	-0.013129	0.008434	-0.010796	-0.001568	
ADI	0.006050	0.010801	-0.010972	0.002495	-0.011887	-0.010525	
ADM	0.007115	-0.011021	0.006709	0.000607	-0.002010	-0.008056	

ticker	X20201230	X20201231
AA	0.039099	0.002172
ABM	-0.002646	0.005565
ABT	-0.002303	0.012129
ADI	0.010522	0.008361
ADM	0.009259	0.007367

[5 rows x 5279 columns]

1.3 3. Sector Data

[34]:
```
# S&P 500 sectors
sectors_sp500 = pd.read_csv(DATA_PATH / "Sectors/Sectors_SP500_YahooNWikipedia.
↪csv")
print(f"S&P 500 sectors shape: {sectors_sp500.shape}")
sectors_sp500.head(10)
```

```
S&P 500 sectors shape: (427, 3)
```

[34]:

	Ticker	Sector_Wikipedia	Sector_Yahoo
0	A	Health_Care	Information_Technology
1	AAP	Consumer_Discretionary	Consumer_Discretionary
2	AAPL	Information_Technology	Information_Technology
3	ABBV	Health_Care	Health_Care
4	ABC	Health_Care	Health_Care
5	ABMD	Health_Care	Health_Care
6	ABT	Health_Care	Health_Care
7	ACN	Information_Technology	Information_Technology
8	ADBE	Information_Technology	Information_Technology
9	ADI	Information_Technology	Information_Technology

[35]:
```
# Column names
print("Columns:", sectors_sp500.columns.tolist())
```

```
Columns: ['Ticker', 'Sector_Wikipedia', 'Sector_Yahoo']
```

[36]:
```
# Unique sectors
print("\nUnique sectors:")
if 'Sector' in sectors_sp500.columns:
    print(sectors_sp500['Sector'].value_counts())
elif 'GICS Sector' in sectors_sp500.columns:
    print(sectors_sp500['GICS Sector'].value_counts())
else:
    print(sectors_sp500.iloc[:, 1:].head())
```

```
Unique sectors:
         Sector_Wikipedia            Sector_Yahoo
0             Health_Care  Information_Technology
1  Consumer_Discretionary  Consumer_Discretionary
2  Information_Technology  Information_Technology
3             Health_Care             Health_Care
4             Health_Care             Health_Care
```

[37]:
```
# S&P 1500 sectors
sectors_sp1500 = pd.read_csv(DATA_PATH / "Sectors/Sectors_SP1500.csv")
print(f"S&P 1500 sectors shape: {sectors_sp1500.shape}")
sectors_sp1500.head()
```

```
S&P 1500 sectors shape: (1459, 4)
```

[37]:

	SPY	0	SPY.1	SPY.2
0	XLK	1	A	Information_Technology
1	XLB	2	AA	Materials
2	XLY	3	AAN	Consumer_Discretionary
3	XLI	4	AAON	Industrials
4	XLY	3	AAP	Consumer_Discretionary

1.4 4. Filter Energy Sector Stocks

[38]:
```
# Find energy stocks
sector_col = [c for c in sectors_sp500.columns if 'sector' in c.lower()]
ticker_col = [c for c in sectors_sp500.columns if 'symbol' in c.lower() or␣
 ↪'ticker' in c.lower()]

print(f"Sector column: {sector_col}")
print(f"Ticker column: {ticker_col}")
```

```
Sector column: ['Sector_Wikipedia', 'Sector_Yahoo']
Ticker column: ['Ticker']
```

[39]:
```
# Extract energy tickers
if sector_col and ticker_col:
    energy_mask = sectors_sp500[sector_col[0]].str.contains('Energy',␣
 ↪case=False, na=False)
    energy_tickers = sectors_sp500.loc[energy_mask, ticker_col[0]].tolist()
    print(f"Energy sector stocks: {len(energy_tickers)}")
    print(energy_tickers)
```

```
Energy sector stocks: 25
['APA', 'COG', 'COP', 'CVX', 'DVN', 'EOG', 'FTI', 'HAL', 'HES', 'HFC', 'HP',
'KMI', 'MPC', 'MRO', 'NBL', 'NOV', 'OKE', 'OXY', 'PSX', 'PXD', 'SLB', 'VLO',
'WMB', 'XEC', 'XOM']
```

[40]:
```
# Check overlap with returns data
if 'energy_tickers' in dir():
    available_energy = [t for t in energy_tickers if t in returns_df.index]
    print(f"Energy tickers in returns data: {len(available_energy)} /␣
 ↪{len(energy_tickers)}")
    print(available_energy)
```

```
Energy tickers in returns data: 14 / 25
['APA', 'COG', 'DVN', 'EOG', 'HAL', 'HP', 'MRO', 'OKE', 'OXY', 'PXD', 'SLB',
'VLO', 'WMB', 'XOM']
```

1.5 5. Summary Statistics

[41]:
```
# Basic stats for a few tickers
sample_tickers = returns_df.index[:5].tolist()
returns_df.loc[sample_tickers].T.describe()
```

[41]:

ticker	AA	ABM	ABT	ADI	ADM
count	5278.000000	5279.000000	5279.000000	5279.000000	5279.000000
mean	0.000218	0.000579	0.000569	0.000664	0.000580
std	0.028507	0.021039	0.015357	0.026674	0.019577
min	-0.210718	-0.185499	-0.161375	-0.166149	-0.168285
25%	-0.013801	-0.008982	-0.006792	-0.011444	-0.008699
50%	0.000000	0.000524	0.000424	0.000398	0.000664
75%	0.013820	0.010397	0.008247	0.011628	0.010010
max	0.282187	0.237489	0.124664	0.229455	0.173378

[42]:
```
# Check if returns are in percentage or decimal form
print("Sample return values:")
print(returns_df.iloc[0, :10].values)
print(f"\nMean of first ticker: {returns_df.iloc[0].mean():.6f}")
print(f"Std of first ticker: {returns_df.iloc[0].std():.6f}")
```

```
Sample return values:
[-0.024849  0.004633  0.057648 -0.013081 -0.002946 -0.002954 -0.005926
 -0.008942 -0.018045 -0.019908]

Mean of first ticker: 0.000218
Std of first ticker: 0.028507
```

[ ]:
```

```