Financial Ratios and Growth Rates

MGMT 638: Data-Driven Investments: Equity

Kerry Back, Rice University

Factors

- Value
- Momentum
- Quality
    - Profitability
    - Low accruals
    - Low asset growth
    - Low default probability
- Volatility (low vol and/or low idiosyncratic vol)
- Liquidity (high volume)

Data

- closeadj, closeunadj, volume from sep_weekly
- marketcap, pb from weekly
- netinc, equity, assets, ncfo from sf1 where dimension="ARQ"

Financial Statement Variables

- Use trailing 4 quarters:
    - netinc, ncfo = sum of prior 4 quarters
    - equity, assets = average of prior 4 quarters
- Variables:
    - roe = netinc / equity
    - accruals = (netinc - ncfo) / equity
    - agr = % change in assets

Create connection

In [1]:

```
import pandas as pd

from sqlalchemy import create_engine
import pymssql
server = 'fs.rice.edu'
database = 'stocks'
username = 'stocks'
password = '6LAZH1'
string = "mssql+pymssql://" + username + ":" + password + "@" + server + "/" +
conn = create_engine(string).connect()
```

Calculate financial ratios and growth rates

Data from SF1

In [2]:

```
sf1 = pd.read_sql(
    """
    select ticker, datekey, lastupdated, netinc, ncfo, equity, assets
    from sf1
    where dimension='ARQ' and datekey>='2009-01-01' and equity>0 and assets>0
    order by ticker, datekey
    """,
    conn,
    parse_dates=["datekey"]
)
sf1 = sf1.groupby(["ticker", "datekey", "lastupdated"]).last()
sf1 = sf1.droplevel("lastupdated")
sf1 = sf1.reset_index()
```

In [3]:

```
for col in ["netinc", "ncfo"]:
    sf1[col] = sf1.groupby("ticker", group_keys=False)[col].apply(
        lambda x: x.rolling(4).sum()
    )
for col in ["equity", "assets"]:
    sf1[col] = sf1.groupby("ticker", group_keys=False)[col].apply(
        lambda x: x.rolling(4).mean()
    )
sf1["roe"] = sf1.netinc / sf1.equity
sf1["accruals"] = (sf1.netinc - sf1.ncfo) / sf1.equity
sf1["agr"] = sf1.groupby("ticker", group_keys=False)["assets"].pct_change()
sf1 = sf1[["ticker", "datekey", "roe", "accruals", "agr"]].dropna()
```

Returns, volume, momentum, volatility

Data from sep_weekly

In [4]:

```
sep_weekly = pd.read_sql(
    """
    select ticker, date, volume, closeadj, closeunadj, lastupdated
    from sep_weekly
    where date>='2010-01-01'
    order by ticker, date, lastupdated
    """,
    conn,
    parse_dates=["date"]
)
sep_weekly = sep_weekly.groupby(["ticker", "date", "lastupdated"]).last()
sep_weekly = sep_weekly.droplevel("lastupdated")
```

In [5]:

```
sep_weekly["ret"] = sep_weekly.groupby("ticker", group_keys=False).closeadj.p
sep_weekly["annual"] = sep_weekly.groupby("ticker", group_keys=False).closead
sep_weekly["monthly"] = sep_weekly.groupby("ticker", group_keys=False).closea
sep_weekly["mom"] = sep_weekly.groupby("ticker", group_keys=False).apply(
    lambda d: (1+d.annual)/(1+d.monthly) - 1
)
sep_weekly["volatility"] = sep_weekly.groupby("ticker", group_keys=False).ret
    lambda x: x.rolling(26).std()
)
sep_weekly = sep_weekly[["ret", "mom", "volume", "volatility", "closeunadj"]]
sep_weekly = sep_weekly.reset_index()
```

Get marketcap and pb

Data from weekly

In [6]:

```
weekly = pd.read_sql(
    """
    select ticker, date, marketcap, pb, lastupdated
    from weekly
    where date>='2010-01-01' and marketcap>0 and pb>0
    order by ticker, date, lastupdated
    """,
    conn,
    parse_dates=["date"]
)
weekly = weekly.groupby(["ticker", "date", "lastupdated"]).last()
weekly = weekly.droplevel("lastupdated")
weekly = weekly.reset_index()
```

Merge

In [7]:

```
df = weekly.merge(sep_weekly, on=["ticker", "date"], how="inner")
df["year"] = df.date.apply(lambda x: x.isocalendar()[0])
df["week"] = df.date.apply(lambda x: x.isocalendar()[1])
sf1["year"] = sf1.datekey.apply(lambda x: x.isocalendar()[0])
sf1["week"] = sf1.datekey.apply(lambda x: x.isocalendar()[1])
df = df.merge(sf1, on=["ticker", "year", "week"], how="left")
df = df.drop(columns=["year", "week", "datekey"])
```

Fill ratios and growth rates forward

In [8]:

```
for col in ["roe", "accruals", "agr"]:
    df[col] = df.groupby("ticker", group_keys=False)[col].apply(
        lambda x: x.ffill()
    )
```

Add sector data

In [ ]:

```
tickers = pd.read_sql(
    """
    select ticker, sector from tickers
    """,
    conn
)
df = df.merge(tickers, on="ticker")
```

Save today's data

In [ ]:

```
today = df[df.date==df.date.max()]
today.to_csv("../../today.csv")
```

Shift weekly features forward

In [9]:
```
for col in ["pb", "mom", "volume", "volatility", "marketcap", "closeunadj"]:
    df[col] = df.groupby("ticker", group_keys=False)[col].shift()
```

Filter to small caps and exclude penny stocks

In [10]:

```
df = df[df.closeunadj>5]
df = df.dropna()
df["rnk"] = df.groupby("date", group_keys=False).marketcap.rank(
    ascending=False,
    method="first"
)
df = df[(df.rnk>1000) & (df.rnk<=3000)]
df = df.drop(columns=["closeunadj", "rnk"])
df = df.sort_values(by=["date", "ticker"])
```

Save data

In [11]:
```
df.to_csv("../../backtest_data.csv", index=False)
```

In [12]:
```
df.head()
```

Out[12]:

	ticker	date	marketcap	pb	ret	mom	volume	volatili
1179	AACC	2011-01-14	188.3	1.4	-0.014634	-0.184615	2.078000e+04	0.0714
2041	AAI	2011-01-14	1012.1	2.0	0.002677	0.438224	2.775580e+06	0.1284
2111	AAIC	2011-01-14	189.3	1.0	-0.010119	0.684547	3.466000e+04	0.0485
4529	AAON	2011-01-14	479.4	4.2	0.007778	0.528685	2.817291e+05	0.0449
7521	AATC	2011-01-14	63.3	1.4	-0.013960	0.008216	6.800000e+03	0.0497