14) Sentiment Analysis: StockTwits Trader Mood

Vitor Kamada

July 2018

Reference

Tables, Graphics, and Figures from

https://www.quantopian.com/tutorials

TUTORIAL 1, LESSONS: 1, 2, 3, and 4

Quantopian's Research Environment

```
from quantopian.research import prices, symbols

# Pandas library: https://pandas.pydata.org/
import pandas as pd

# Query historical pricing data for AAPL
aapl_close = prices(
    assets=symbols('AAPL'),
    start='2013-01-01',
    end='2016-01-01',
)
```

Compute 20 and 50 day Moving Averages

```
aapl_sma20 = aapl_close.rolling(20).mean()
aapl_sma50 = aapl_close.rolling(50).mean()

# Combine results into a pandas DataFrame and plot
pd.DataFrame({
    'AAPL': aapl_close,
    'SMA20': aapl_sma20,
    'SMA50': aapl_sma50
}).plot(
    title='AAPL Close Price / SMA Crossover'
);
```

AAPL Close Price / SMA Crossover

Pipeline Imports

```
from quantopian.research import run_pipeline
from quantopian.pipeline import Pipeline
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.data.psychsignal import stocktwits

# Pipeline definition
def make_pipeline():

    returns = Returns(window_length=2)
    sentiment = stocktwits.bull_minus_bear.latest
    msg_volume = stocktwits.total_scanned_messages.latest

    return Pipeline(
        columns={
            'daily_returns': returns,
            'sentiment': sentiment,
            'msg_volume': msg_volume,
        },
    )
```

Pipeline Execution

```
data_output = run_pipeline(
    make_pipeline(),
    start_date=period_start,
    end_date=period_end
)

# Filter results for AAPL
aapl_output = data_output.xs(
    symbols('AAPL'),
    level=1
)

# Plot results for AAPL
aapl_output.plot(subplots=True);
```

StockTwits Trader Mood

Import Pipeline Class and Datasets

```
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data import USEquityPricing
from quantopian.pipeline.data.psychsignal import stocktwits

# Import built-in moving average calculation
from quantopian.pipeline.factors import SimpleMovingAverage

# Import built-in trading universe
from quantopian.pipeline.filters import QTradableStocksUS
```

Make Pipeline

```
def make_pipeline():
    # Create a reference to our trading universe
    base_universe = QTradableStocksUS()

    # Get latest closing price
    close_price = USEquityPricing.close.latest

    # Calculate 3 day average of bull_minus_bear scores
    sentiment_score = SimpleMovingAverage(
        inputs=[stocktwits.bull_minus_bear],
        window_length=3,
    )
    return Pipeline(
        columns={
            'close_price': close_price,
            'sentiment_score': sentiment_score,
        },
        screen=base_universe
    )
```

Run Pipeline

```
from quantopian.research import run_pipeline

# Execute pipeline created by make_pipeline
# between start_date and end_date
pipeline_output = run_pipeline(
    make_pipeline(),
    start_date='2013-01-01',
    end_date='2013-12-31'
)

# Display last 10 rows
pipeline_output.tail(10)
```

Pipeline Output

		close_price	sentiment_score
2013-12-31 00:00:00+00:00	Equity(43721 [SCTY])	57.32	-0.176667
	Equity(43919 [LMCA])	146.22	0.000000
	Equity(43981 [NCLH])	35.25	-0.700000
	Equity(44053 [TPH])	19.33	0.333333
	Equity(44060 [ZTS])	32.68	0.000000
	Equity(44089 [BCC])	29.66	1.000000
	Equity(44102 [XONE])	60.50	0.396667
	Equity(44158 [XOOM])	27.31	-0.160000
	Equity(44249 [APAM])	64.53	0.000000
	Equity(44270 [SSNI])	21.05	0.423333

Pipeline Definition

```
def make_pipeline():

    base_universe = QTradableStocksUS()

    sentiment_score = SimpleMovingAverage(
        inputs=[stocktwits.bull_minus_bear],
        window_length=3,
    )

    # Create filter for top 350 and bottom 350
    # assets based on their sentiment scores
    top_bottom_scores = (
        sentiment_score.top(350) | sentiment_score.bottom(350)
    )
```

Pipeline Definition (Continuation)

```
    return Pipeline(
        columns={
            'sentiment_score': sentiment_score,
        },
        # Set screen as the intersection between our filter
        # and trading universe
        screen=(
            base_universe
            & top_bottom_scores
        )
    )
```

Run Pipeline over 3 year

```
from quantopian.research import run_pipeline

# Specify a time range to evaluate
period_start = '2013-01-01'
period_end = '2016-01-01'

# Execute pipeline over evaluation period
pipeline_output = run_pipeline(
    make_pipeline(),
    start_date=period_start,
    end_date=period_end
)
```

Import Prices Function

```
from quantopian.research import prices

# Get list of unique assets from the pipeline output
asset_list = pipeline_output.index.levels[1].unique()

# Query pricing data for all assets present during
# evaluation period
asset_prices = prices(
    asset_list,
    start=period_start,
    end=period_end
)
```

Alphalens - Factor Analysis Tool

```
import alphalens as al

# Get asset forward returns and quantile classification
# based on sentiment scores
factor_data = al.utils.get_clean_factor_and_forward_returns(
    factor=pipeline_output['sentiment_score'],
    prices=asset_prices,
    quantiles=2,
    periods=(1,5,10),
)

# Display first 5 rows
factor_data.head(5)
```

Factor Data

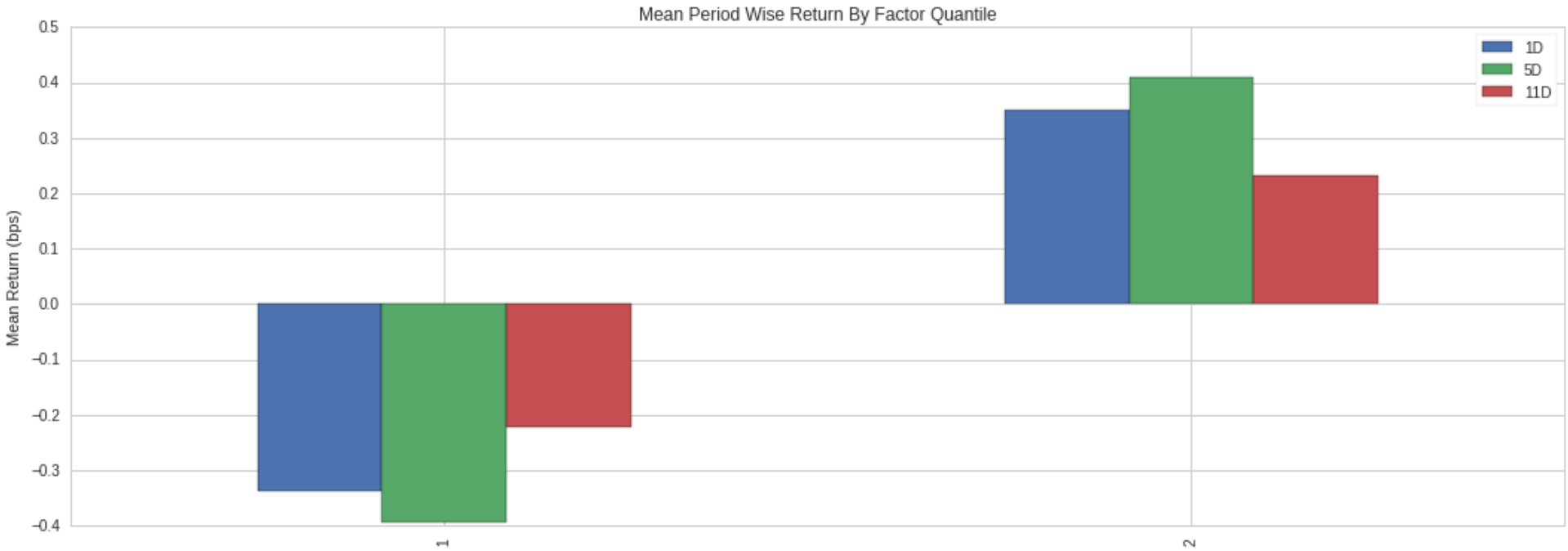
		1D	5D	11D	factor	factor_quantile
date	**asset**					
2013-01-02 00:00:00+00:00	**Equity(52 [ABM])**	0.004430	0.004430	0.004430	2.560000	2
	Equity(114 [ADBE])	-0.015389	0.008086	-0.012259	-1.896667	1
	Equity(166 [AES])	-0.006368	-0.008104	-0.005403	-2.630000	1
	Equity(209 [AM])	0.001801	-0.022995	-0.038365	2.370000	2
	Equity(337 [AMAT])	-0.002525	-0.014339	0.007575	2.370000	2

Mean Return by Factor Quantile

```
mean_return_by_q, std_err_by_q = al.performance.mean_return_by_quantile(factor_data)

# Plot mean returns by quantile and holding period
# over evaluation time range
al.plotting.plot_quantile_returns_bar(
    mean_return_by_q.apply(
        al.utils.rate_of_return,
        axis=0,
        args=('1D',)
    )
);
```

Mean Period Wise Return by Factor Quantile



Portfolio Cumulative Return (5D Fwd Period)

```
# Calculate factor-weighted long-short portfolio returns
ls_factor_returns = al.performance.factor_returns(factor_data)

# Plot cumulative returns for 5 day holding period
al.plotting.plot_cumulative_returns(ls_factor_returns['5D'], '5D');
```