02_backtest_with_pf_optimization

September 29, 2021

1 MeanReversion backtest with Portfolio Optimization

In the chapter 04, we introduced `zipline` to simulate the computation of alpha factors from trailing cross-sectional market, fundamental, and alternative data.

Now we will exploit the alpha factors to derive and act on buy and sell signals using the custom MeanReversion factor developed in the last chapter.

1.1 Imports

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import sys
from pytz import UTC
import logbook

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from logbook import (NestedSetup, NullHandler, Logger,
                     StreamHandler, StderrHandler,
                     INFO, WARNING, DEBUG, ERROR)

from zipline import run_algorithm
from zipline.api import (attach_pipeline,
                         date_rules,
                         time_rules,
                         get_datetime,
                         order_target_percent,
                         pipeline_output,
                         record, schedule_function,
                         get_open_orders,
                         calendars,
                         set_commission,
                         set_slippage)
```

```
from zipline.finance import commission, slippage
from zipline.pipeline import Pipeline, CustomFactor
from zipline.pipeline.factors import Returns, AverageDollarVolume

from pypfopt.efficient_frontier import EfficientFrontier
from pypfopt import risk_models, objective_functions
from pypfopt import expected_returns
from pypfopt.exceptions import OptimizationError

from pyfolio.utils import extract_rets_pos_txn_from_zipline
```

[3]:
```
sns.set_style('whitegrid')
```

1.2 Logging Setup

[4]:
```
# setup stdout logging
format_string = '[{record.time: %H:%M:%S.%f}]: {record.level_name}: {record.
 ↪message}'
zipline_logging = NestedSetup([NullHandler(level=DEBUG),
                               StreamHandler(sys.stdout,␣
 ↪format_string=format_string, level=INFO),
                               StreamHandler(sys.stdout,␣
 ↪format_string=format_string, level=WARNING),
                               StreamHandler(sys.stderr, level=ERROR)])
zipline_logging.push_application()
log = Logger('Algorithm')
```

1.3 Algo Settings

[5]:
```
# Settings
MONTH = 21
YEAR = 12 * MONTH
N_LONGS = 50
N_SHORTS = 50
MIN_POS = 5
VOL_SCREEN = 1000
```

[6]:
```
start = pd.Timestamp('2013-01-01', tz=UTC)
end = pd.Timestamp('2017-01-01', tz=UTC)
capital_base = 1e7
```

1.4 Mean Reversion Factor

[7]:
```
class MeanReversion(CustomFactor):
    """Compute ratio of latest monthly return to 12m average,
       normalized by std dev of monthly returns"""
    inputs = [Returns(window_length=MONTH)]
    window_length = YEAR

    def compute(self, today, assets, out, monthly_returns):
        df = pd.DataFrame(monthly_returns)
        factor = df.iloc[-1].sub(df.mean()).div(df.std())
        out[:] = factor
```

1.5 Create Pipeline

The Pipeline created by the `compute_factors()` method returns a table with a long and a short column for the 25 stocks with the largest negative and positive deviations of their last monthly return from its annual average, normalized by the standard deviation. It also limited the universe to the 500 stocks with the highest average trading volume over the last 30 trading days.

[8]:
```
def compute_factors():
    """Create factor pipeline incl. mean reversion,
        filtered by 30d Dollar Volume; capture factor ranks"""
    mean_reversion = MeanReversion()
    dollar_volume = AverageDollarVolume(window_length=30)
    return Pipeline(columns={'longs'  : mean_reversion.bottom(N_LONGS),
                             'shorts' : mean_reversion.top(N_SHORTS),
                             'ranking': mean_reversion.rank(ascending=False)},
                    screen=dollar_volume.top(VOL_SCREEN))
```

`Before_trading_start()` ensures the daily execution of the pipeline and the recording of the results, including the current prices.

[9]:
```
def before_trading_start(context, data):
    """Run factor pipeline"""
    context.factor_data = pipeline_output('factor_pipeline')
    record(factor_data=context.factor_data.ranking)
    assets = context.factor_data.index
    record(prices=data.current(assets, 'price'))
```

1.6 Set up Rebalancing

The new `rebalance()` method submits trade orders to the `exec_trades()` method for the assets flagged for long and short positions by the pipeline with equal positive and negative weights.

It also divests any current holdings that are no longer included in the factor signals:

[10]:
```
def exec_trades(data, positions):
    """Place orders for assets using target portfolio percentage"""
    for asset, target_percent in positions.items():
        if data.can_trade(asset) and not get_open_orders(asset):
            order_target_percent(asset, target_percent)
```

[11]:
```
def rebalance(context, data):
    """Compute long, short and obsolete holdings; place orders"""

    factor_data = context.factor_data
    assets = factor_data.index

    longs = assets[factor_data.longs]
    shorts = assets[factor_data.shorts]

    divest = context.portfolio.positions.keys() - longs.union(shorts)
    exec_trades(data, positions={asset: 0 for asset in divest})
    log.info('{} | {:11,.0f}'.format(get_datetime().date(),
                                     context.portfolio.portfolio_value))

    # get price history
    prices = data.history(assets, fields='price',
                          bar_count=252+1, # for 1 year of returns
                          frequency='1d')

    # get optimal weights if sufficient candidates
    if len(longs) > MIN_POS and len(shorts) > MIN_POS:
        try:
            long_weights = optimize_weights(prices.loc[:, longs])
            short_weights = optimize_weights(prices.loc[:, shorts], short=True)

            exec_trades(data, positions=long_weights)
            exec_trades(data, positions=short_weights)
        except Exception as e:
            log.warn('{} {}'.format(get_datetime().date(), e))
    # exit remaining positions
    divest_pf = {asset: 0 for asset in context.portfolio.positions.keys()}
    exec_trades(data, positions=divest_pf)
```

1.7 Optimize Portfolio Weights

[12]:
```
def optimize_weights(prices, short=False):

    returns = expected_returns.mean_historical_return(
        prices=prices, frequency=252)
    cov = risk_models.sample_cov(prices=prices, frequency=252)
```

```
    # get weights that maximize the Sharpe ratio
    ef = EfficientFrontier(expected_returns=returns,
                           cov_matrix=cov,
                           weight_bounds=(0, 1),
                           solver='SCS')
    ef.max_sharpe()
    if short:
        return {asset: -weight for asset, weight in ef.clean_weights().items()}
    else:
        return ef.clean_weights()
```

1.8 Initialize Backtest

The `rebalance()` method runs according to `date_rules` and `time_rules` set by the `schedule_function()` utility at the beginning of the week, right after `market_open` as stipulated by the built-in `US_EQUITIES` calendar (see docs for details on rules).

You can also specify a trade commission both in relative terms and as a minimum amount. There is also an option to define slippage, which is the cost of an adverse change in price between trade decision and execution

[13]:
```
def initialize(context):
    """Setup: register pipeline, schedule rebalancing,
        and set trading params"""
    attach_pipeline(compute_factors(), 'factor_pipeline')
    schedule_function(rebalance,
                      date_rules.week_start(),
                      time_rules.market_open(),
                      calendar=calendars.US_EQUITIES)

    set_commission(us_equities=commission.PerShare(cost=0.00075,␣
 ↪min_trade_cost=.01))
    set_slippage(us_equities=slippage.VolumeShareSlippage(volume_limit=0.0025,␣
 ↪price_impact=0.01))
```

1.9 Run Algorithm

The algorithm executes upon calling the `run_algorithm()` function and returns the backtest performance `DataFrame`.

[14]:
```
backtest = run_algorithm(start=start,
                         end=end,
                         initialize=initialize,
                         before_trading_start=before_trading_start,
                         bundle='quandl',
                         capital_base=capital_base)
```

```
[ 19:49:55.661120]: INFO: 2013-01-07 |  10,000,000
[ 19:50:02.961404]: INFO: 2013-01-14 |   9,922,439
[ 19:50:06.101723]: INFO: 2013-01-22 |   9,913,612
[ 19:50:09.110052]: INFO: 2013-01-28 |   9,968,656
[ 19:50:12.293743]: INFO: 2013-02-04 |  10,079,391
[ 19:50:15.636407]: INFO: 2013-02-11 |  10,103,535
[ 19:50:19.025936]: INFO: 2013-02-19 |   9,771,077
[ 19:50:22.454159]: INFO: 2013-02-25 |  10,061,824
[ 19:50:26.025560]: INFO: 2013-03-04 |  10,162,312
[ 19:50:29.300896]: INFO: 2013-03-11 |   9,563,302
[ 19:50:33.294339]: INFO: 2013-03-18 |   9,311,865
[ 19:50:36.611748]: INFO: 2013-03-25 |   9,064,119
[ 19:50:39.810862]: INFO: 2013-04-01 |   8,643,936
[ 19:50:43.357770]: INFO: 2013-04-08 |   8,721,979
[ 19:50:47.126182]: INFO: 2013-04-15 |   8,552,107
[ 19:50:51.070580]: INFO: 2013-04-22 |   8,186,845
[ 19:50:55.027566]: INFO: 2013-04-29 |   9,080,169
[ 19:50:58.655614]: INFO: 2013-05-06 |   8,277,424
[ 19:51:02.617136]: INFO: 2013-05-13 |   8,018,122
[ 19:51:06.524561]: INFO: after split: asset: Equity(189 [AOS]), amount: 62678,
cost_basis: 36.54, last_sale_price: 79.9
[ 19:51:06.524914]: INFO: returning cash: 0.0
[ 19:51:07.067627]: INFO: 2013-05-20 |   7,805,480
[ 19:51:11.239217]: INFO: 2013-05-28 |   6,806,488
[ 19:51:15.113238]: INFO: 2013-06-03 |   7,034,540
[ 19:51:19.219741]: INFO: 2013-06-10 |   7,315,434
[ 19:51:23.416837]: INFO: 2013-06-17 |   7,620,513
[ 19:51:27.623110]: INFO: 2013-06-24 |   6,403,342
[ 19:51:31.930314]: INFO: 2013-07-01 |   5,958,489
[ 19:51:36.417751]: INFO: 2013-07-08 |   5,427,714
[ 19:51:42.579167]: INFO: 2013-07-15 |   5,956,660
[ 19:51:47.402413]: INFO: 2013-07-22 |   6,237,875
[ 19:51:51.883184]: INFO: 2013-07-29 |   6,059,791
[ 19:51:55.834086]: INFO: 2013-08-05 |   6,454,247
[ 19:52:00.032541]: INFO: 2013-08-12 |   6,968,293
[ 19:52:04.023979]: INFO: 2013-08-19 |   5,543,339
[ 19:52:07.863710]: INFO: 2013-08-26 |   6,377,059
[ 19:52:12.136057]: INFO: 2013-09-03 |   6,426,491
[ 19:52:15.910429]: INFO: 2013-09-09 |   7,392,591
[ 19:52:20.035565]: INFO: 2013-09-16 |   7,315,055
[ 19:52:25.921373]: INFO: 2013-09-23 |   7,812,821
[ 19:52:30.754606]: INFO: 2013-09-30 |   6,988,949
[ 19:52:34.801378]: INFO: 2013-10-07 |   6,107,561
[ 19:52:38.671079]: INFO: 2013-10-14 |   6,255,426
[ 19:52:42.044195]: INFO: 2013-10-21 |   7,069,346
[ 19:52:45.102026]: INFO: 2013-10-28 |   7,059,135
[ 19:52:48.383666]: INFO: 2013-11-04 |   7,314,434
[ 19:52:51.121046]: INFO: 2013-11-11 |   6,430,076
```

```
[ 19:52:54.211109]: INFO: 2013-11-18 |   6,851,219
[ 19:52:57.926595]: INFO: 2013-11-25 |   6,795,541
[ 19:53:00.937046]: INFO: 2013-12-02 |   7,138,853
[ 19:53:03.877547]: INFO: 2013-12-09 |   6,881,797
[ 19:53:07.190402]: INFO: 2013-12-16 |   6,543,537
[ 19:53:10.698781]: INFO: 2013-12-23 |   7,439,189
[ 19:53:14.008429]: INFO: 2013-12-30 |   7,441,863
[ 19:53:17.436756]: INFO: 2014-01-06 |   7,369,341
[ 19:53:20.990705]: INFO: 2014-01-13 |   7,079,246
[ 19:53:26.130491]: INFO: 2014-01-21 |   6,746,705
[ 19:53:29.472657]: INFO: 2014-01-27 |   6,519,510
[ 19:53:32.492814]: INFO: 2014-02-03 |   6,583,762
[ 19:53:35.400119]: INFO: 2014-02-10 |   7,061,039
[ 19:53:38.446365]: INFO: 2014-02-18 |   7,884,748
[ 19:53:41.424621]: INFO: 2014-02-24 |   7,668,098
[ 19:53:43.864251]: INFO: 2014-03-03 |   8,758,089
[ 19:53:46.280781]: INFO: 2014-03-10 |   8,835,178
[ 19:53:48.517062]: INFO: 2014-03-17 |   8,778,877
[ 19:53:51.167632]: INFO: 2014-03-24 |   8,370,815
[ 19:53:54.435749]: INFO: 2014-03-31 |   8,441,825
[ 19:53:57.105086]: INFO: 2014-04-07 |   7,674,570
[ 19:54:00.153841]: INFO: 2014-04-14 |   7,078,187
[ 19:54:02.991366]: INFO: 2014-04-21 |   7,688,825
[ 19:54:05.843967]: INFO: 2014-04-28 |   7,589,755
[ 19:54:08.687119]: INFO: 2014-05-05 |   7,968,224
[ 19:54:11.282235]: INFO: 2014-05-12 |   8,181,812
[ 19:54:13.907430]: INFO: 2014-05-19 |   7,101,661
[ 19:54:16.504037]: INFO: 2014-05-27 |   7,566,057
[ 19:54:19.270873]: INFO: 2014-06-02 |   7,396,299
[ 19:54:21.679662]: INFO: 2014-06-09 |   7,720,697
[ 19:54:24.253035]: INFO: 2014-06-16 |   7,469,853
[ 19:54:27.093635]: INFO: 2014-06-23 |   7,480,065
[ 19:54:29.915811]: INFO: 2014-06-30 |   7,816,351
[ 19:54:32.576913]: INFO: 2014-07-07 |   8,052,165
[ 19:54:35.525930]: INFO: 2014-07-14 |   7,970,295
[ 19:54:40.430470]: INFO: 2014-07-21 |   8,376,080
[ 19:54:43.443027]: INFO: 2014-07-28 |   8,244,746
[ 19:54:46.840045]: INFO: 2014-08-04 |   8,341,610
[ 19:54:49.967235]: INFO: 2014-08-11 |   8,860,966
[ 19:54:53.727961]: INFO: 2014-08-18 |   9,366,271
[ 19:54:56.397549]: INFO: 2014-08-25 |   9,410,676
[ 19:54:58.717063]: INFO: 2014-09-02 |   9,513,942
[ 19:55:00.885053]: INFO: 2014-09-08 |   8,991,093
[ 19:55:03.514441]: INFO: 2014-09-15 |   9,269,792
[ 19:55:06.189510]: INFO: 2014-09-22 |   9,231,904
[ 19:55:08.703271]: INFO: 2014-09-29 |   8,066,988
[ 19:55:12.292468]: INFO: 2014-10-06 |   6,258,677
[ 19:55:15.435277]: INFO: 2014-10-13 |   4,441,940
```

```
[ 19:55:18.151838]: INFO: 2014-10-20 |   5,541,794
[ 19:55:20.750519]: INFO: 2014-10-27 |   6,187,174
[ 19:55:23.280320]: INFO: 2014-11-03 |   7,400,147
[ 19:55:25.680151]: INFO: 2014-11-10 |   7,287,218
[ 19:55:28.012422]: INFO: 2014-11-17 |   7,684,538
[ 19:55:30.227017]: INFO: 2014-11-24 |   8,245,927
[ 19:55:32.100307]: INFO: 2014-12-01 |   7,381,184
[ 19:55:34.806464]: INFO: 2014-12-08 |   7,766,157
[ 19:55:36.782542]: INFO: 2014-12-15 |   7,681,728
[ 19:55:38.733397]: INFO: 2014-12-22 |   8,411,340
[ 19:55:40.649713]: INFO: 2014-12-29 |   8,317,332
[ 19:55:42.647684]: INFO: 2015-01-05 |   7,382,470
[ 19:55:45.148846]: INFO: 2015-01-12 |   7,575,066
[ 19:55:49.376307]: INFO: 2015-01-20 |   7,916,220
[ 19:55:51.376484]: INFO: 2015-01-26 |   8,261,115
[ 19:55:54.002615]: INFO: 2015-02-02 |   7,365,550
[ 19:55:56.778925]: INFO: 2015-02-09 |   7,536,988
[ 19:55:58.952506]: INFO: 2015-02-17 |   7,765,648
[ 19:56:00.905802]: INFO: 2015-02-23 |   7,499,103
[ 19:56:03.030776]: INFO: 2015-03-02 |   6,865,120
[ 19:56:05.550942]: INFO: 2015-03-09 |   6,105,947
[ 19:56:08.617395]: INFO: 2015-03-16 |   5,998,193
[ 19:56:12.668770]: INFO: 2015-03-23 |   6,102,339
[ 19:56:16.999381]: INFO: 2015-03-30 |   5,852,169
[ 19:56:20.316627]: INFO: 2015-04-06 |   5,839,392
[ 19:56:24.030473]: INFO: 2015-04-13 |   6,002,854
[ 19:56:25.876552]: INFO: 2015-04-20 |   5,858,690
[ 19:56:27.481938]: INFO: 2015-04-27 |   5,578,378
[ 19:56:29.033304]: INFO: 2015-05-04 |   5,727,009
[ 19:56:30.560601]: INFO: 2015-05-11 |   5,876,390
[ 19:56:32.108828]: INFO: 2015-05-18 |   5,926,198
[ 19:56:33.658631]: INFO: 2015-05-26 |   5,870,281
[ 19:56:34.898588]: INFO: 2015-06-01 |   6,357,525
[ 19:56:36.288508]: INFO: 2015-06-08 |   6,563,835
[ 19:56:37.974627]: INFO: 2015-06-15 |   6,356,822
[ 19:56:39.385081]: INFO: 2015-06-22 |   7,143,691
[ 19:56:40.777906]: INFO: 2015-06-29 |   6,302,243
[ 19:56:41.858567]: INFO: 2015-07-06 |   6,485,967
[ 19:56:43.265941]: INFO: 2015-07-13 |   6,928,867
[ 19:56:45.021742]: INFO: 2015-07-20 |   7,051,650
[ 19:56:48.614537]: INFO: 2015-07-27 |   7,044,274
[ 19:56:50.681199]: INFO: 2015-08-03 |   7,323,573
[ 19:56:52.769434]: INFO: 2015-08-10 |   7,568,364
[ 19:56:55.288387]: INFO: 2015-08-17 |   7,658,959
[ 19:56:57.335410]: INFO: 2015-08-24 |   5,970,305
[ 19:56:59.408318]: INFO: 2015-08-31 |   6,637,912
[ 19:57:01.459935]: INFO: 2015-09-08 |   7,096,752
[ 19:57:03.360521]: INFO: 2015-09-14 |   7,078,391
```

```
[ 19:57:05.419445]: INFO: 2015-09-21 |   7,109,001
[ 19:57:07.657716]: INFO: 2015-09-28 |   6,892,548
[ 19:57:09.713700]: INFO: 2015-10-05 |   7,432,601
[ 19:57:11.782813]: INFO: 2015-10-12 |   8,110,083
[ 19:57:14.358990]: INFO: 2015-10-19 |   8,164,760
[ 19:57:16.437731]: INFO: 2015-10-26 |   8,446,339
[ 19:57:18.547980]: INFO: 2015-11-02 |   9,391,155
[ 19:57:20.661324]: INFO: 2015-11-09 |   8,829,317
[ 19:57:22.917628]: INFO: 2015-11-16 |   8,434,693
[ 19:57:24.462049]: INFO: after split: asset: Equity(406 [BOFI]), amount: 32844,
cost_basis: 22.22, last_sale_price: 83.29
[ 19:57:24.462437]: INFO: returning cash: 0.0
[ 19:57:25.152353]: INFO: 2015-11-23 |   9,203,580
[ 19:57:27.240601]: INFO: 2015-11-30 |   9,555,485
[ 19:57:29.486857]: INFO: 2015-12-07 |   9,152,902
[ 19:57:31.896704]: INFO: 2015-12-14 |   8,799,672
[ 19:57:34.736330]: INFO: 2015-12-21 |   9,181,258
[ 19:57:36.855101]: INFO: 2015-12-28 |   9,167,269
[ 19:57:38.942283]: INFO: 2016-01-04 |   8,890,359
[ 19:57:41.091240]: INFO: 2016-01-11 |   7,867,946
[ 19:57:43.371758]: INFO: 2016-01-19 |   7,562,504
[ 19:57:47.218014]: INFO: 2016-01-25 |   7,257,377
[ 19:57:48.692725]: WARNING: 2016-01-25 ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
[ 19:57:49.570712]: INFO: 2016-02-01 |   7,359,763
[ 19:57:51.939493]: INFO: 2016-02-08 |   4,641,970
[ 19:57:54.339881]: INFO: 2016-02-16 |   5,248,988
[ 19:57:56.978133]: INFO: 2016-02-22 |   7,136,137
[ 19:57:59.381385]: INFO: 2016-02-29 |   6,489,040
[ 19:58:01.786348]: INFO: 2016-03-07 |   7,469,882
[ 19:58:04.176099]: INFO: 2016-03-14 |   8,954,946
[ 19:58:06.530112]: INFO: 2016-03-21 |   9,471,995
[ 19:58:08.022997]: WARNING: 2016-03-21 ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
[ 19:58:08.684584]: INFO: 2016-03-28 |   9,328,094
[ 19:58:11.034063]: INFO: 2016-04-04 |   9,743,417
[ 19:58:13.371768]: INFO: 2016-04-11 |  10,087,318
[ 19:58:15.804977]: INFO: 2016-04-18 |  10,745,639
[ 19:58:18.945575]: INFO: 2016-04-25 |  10,806,461
[ 19:58:21.691592]: INFO: 2016-05-02 |  11,879,697
[ 19:58:24.474339]: INFO: 2016-05-09 |  12,908,341
[ 19:58:27.379225]: INFO: 2016-05-16 |  12,124,789
[ 19:58:30.108167]: INFO: 2016-05-23 |  10,192,441
[ 19:58:31.905902]: WARNING: 2016-05-23 ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
[ 19:58:32.783601]: INFO: 2016-05-31 |  11,063,885
[ 19:58:35.369845]: INFO: 2016-06-06 |  11,275,450
[ 19:58:38.090709]: INFO: 2016-06-13 |  11,089,536
```

```
[ 19:58:40.802084]: INFO: 2016-06-20 |  11,424,829
[ 19:58:43.010284]: WARNING: 2016-06-20 ('Please check your
objectives/constraints or use a different solver.', 'Solver status: infeasible')
[ 19:58:43.850453]: INFO: 2016-06-27 |  10,402,124
[ 19:58:46.382454]: INFO: 2016-07-05 |  10,926,853
[ 19:58:48.428655]: INFO: 2016-07-11 |  11,208,081
[ 19:58:50.795378]: INFO: 2016-07-18 |  10,898,410
[ 19:58:57.111404]: INFO: 2016-07-25 |  10,671,246
[ 19:58:59.400699]: INFO: 2016-08-01 |  11,244,511
[ 19:59:01.682477]: INFO: 2016-08-08 |  10,974,343
[ 19:59:03.996200]: INFO: 2016-08-15 |  10,499,160
[ 19:59:06.331738]: INFO: 2016-08-22 |  10,103,598
[ 19:59:08.921478]: INFO: 2016-08-29 |   9,995,782
[ 19:59:11.135088]: INFO: 2016-09-06 |   9,708,339
[ 19:59:13.118201]: INFO: 2016-09-12 |  10,138,554
[ 19:59:15.280128]: INFO: 2016-09-19 |   9,888,820
[ 19:59:17.487149]: INFO: 2016-09-26 |  10,384,830
[ 19:59:19.692916]: INFO: 2016-10-03 |   9,870,885
[ 19:59:21.863634]: INFO: 2016-10-10 |   9,007,649
[ 19:59:24.064888]: INFO: 2016-10-17 |   8,642,711
[ 19:59:26.447127]: INFO: 2016-10-24 |   8,723,289
[ 19:59:29.274611]: INFO: 2016-10-31 |   9,861,873
[ 19:59:31.524188]: INFO: 2016-11-07 |  10,512,150
[ 19:59:33.752999]: INFO: 2016-11-14 |   9,782,800
[ 19:59:36.172179]: INFO: 2016-11-21 |   7,302,885
[ 19:59:38.165779]: INFO: 2016-11-28 |   8,179,329
[ 19:59:40.247054]: INFO: 2016-12-05 |   6,719,502
[ 19:59:42.224743]: INFO: 2016-12-12 |   6,428,943
[ 19:59:43.996133]: INFO: 2016-12-19 |   4,324,427
[ 19:59:45.862659]: INFO: 2016-12-27 |   3,500,602
[ 19:59:47.796972]: INFO: Simulated 1008 trading days
first open: 2013-01-02 14:31:00+00:00
last close: 2016-12-30 21:00:00+00:00
```

1.10 Extract pyfolio Inputs

The `extract_rets_pos_txn_from_zipline` utility provided by `pyfolio` extracts the data used to compute performance metrics.

[15]:
```
returns, positions, transactions = extract_rets_pos_txn_from_zipline(backtest)
```

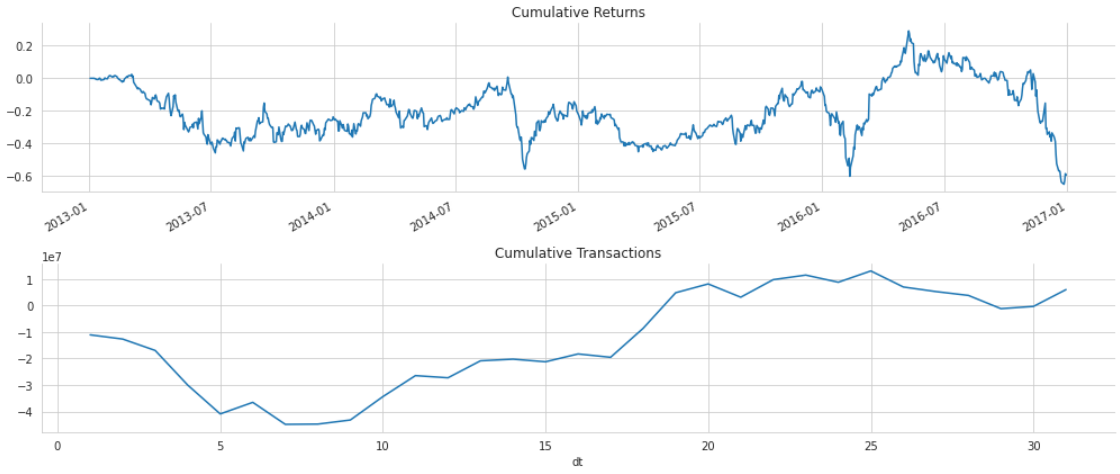
1.11 Persist Results for use with `pyfolio`

[16]:
```
with pd.HDFStore('backtests.h5') as store:
    store.put('returns/pf_opt', returns)
    store.put('transactions/pf_opt', transactions)
```

[17]:
```
with pd.HDFStore('backtests.h5') as store:
    returns_pf = store['returns/pf_opt']
    tx_pf = store['transactions/pf_opt']
    returns_ew = store['returns/equal_weight']
    tx_ew = store['transactions/equal_weight']
```

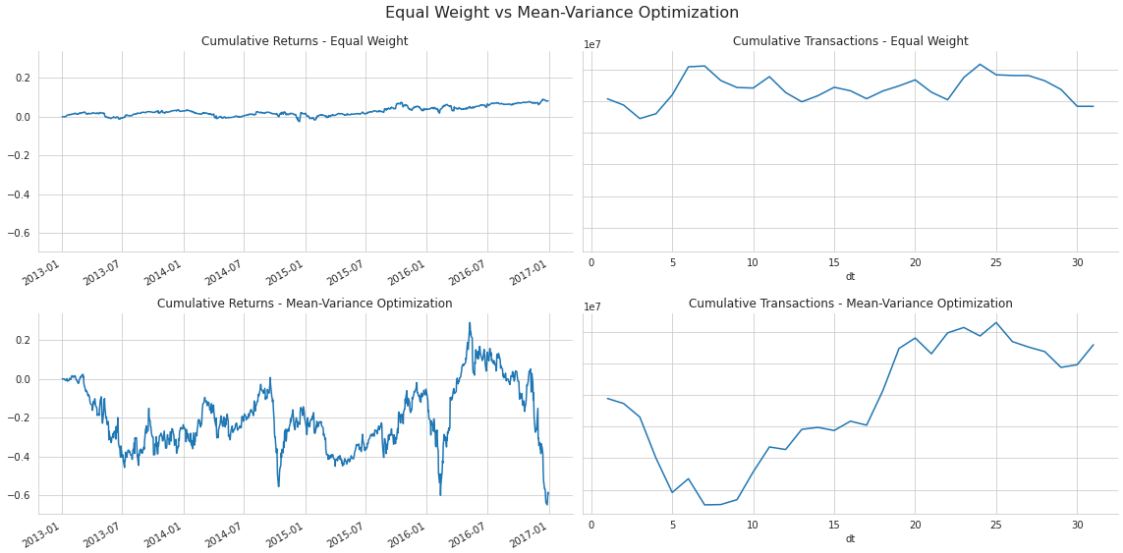
1.12 Plot Results

[18]:
```
fig, axes= plt.subplots(nrows=2, figsize=(14,6))
returns.add(1).cumprod().sub(1).plot(ax=axes[0], title='Cumulative Returns')
transactions.groupby(transactions.dt.dt.day).txn_dollars.sum().cumsum().
 ↪plot(ax=axes[1], title='Cumulative Transactions')
sns.despine()
fig.tight_layout();
```



[19]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16, 8), sharey='col')
returns_ew.add(1).cumprod().sub(1).plot(ax=axes[0][0],
                                        title='Cumulative Returns - Equal␣
 ↪Weight')
returns_pf.add(1).cumprod().sub(1).plot(ax=axes[1][0],
                                        title='Cumulative Returns -␣
 ↪Mean-Variance Optimization')
tx_ew.groupby(tx_ew.dt.dt.day).txn_dollars.sum().cumsum().plot(ax=axes[0][1],
                                                               ␣
 ↪title='Cumulative Transactions - Equal Weight')
tx_pf.groupby(tx_pf.dt.dt.day).txn_dollars.sum().cumsum().plot(ax=axes[1][1],
                                                               ␣
 ↪title='Cumulative Transactions - Mean-Variance Optimization')
fig.suptitle('Equal Weight vs Mean-Variance Optimization', fontsize=16)
sns.despine()
```

```
fig.tight_layout()
fig.subplots_adjust(top=.9)
```



[ ]: