PyAlgoTrade Documentation

发布 ***0.17***

Gabriel Martín Becedillas Ruiz

6月 28, 2017

Contents

目录:

CHAPTER 1

简介

PyAlgoTrade作为事件驱动型的算法交易工具，其支持:

- 从CSV文件中获取历史数据并进行回测。
- 通过使用 *Xignite* 和 *Bitstamp* 的实时数据进行模拟交易。
- 在Bitstamp上进行实战交易。

它可以很方便的在多台计算机之间优化策略。

PyAlgoTrade基于Python2.7开发并依托于以下python库:

- NumPy and SciPy (http://numpy.scipy.org/).：科学计算库
- pytz (http://pytz.sourceforge.net/).
- matplotlib (http://matplotlib.sourceforge.net/)：绘图库
- ws4py (https://github.com/Lawouach/WebSocket-for-Python)：用于得到Bitstamp支持
- tornado (http://www.tornadoweb.org/en/stable/) 用于得到Bitstamp支持.
- tweepy (https://github.com/tweepy/tweepy) 用于得到推特支持.

因此，你需要安装上述扩展库才能使用本扩展库。

你可以通过pip来安装PyAlgoTrade:

```
pip install pyalgotrade
```

译者注:

0.17版 本 已 经 迁 移 到Python3.x上 ， 原 版 只 能 在py27环 境 下 运 行 ， 请 谨 慎 升 级 。 down:PyAlgoTrade0.16_for_py3

2016-2-23:

```
目前，已有国内版本
地址：https://github.com/Yam-cn/pyalgotrade-cn
```

```
基于py2.7，支持A股交易、以及封装了tushare的A股即时行情
项目QQ群：300349971
```

水平有限，如有疏漏，敬请谅解！欢迎一起翻译&debug!

CHAPTER 2

简明教程

本教程的目的是让你能够快速入门PyAlgoTrade。 如之前简介中所述，PyAlgoTrade的目标是帮助你测试股票交易策略。 一般而言，如果你有一个交易策略的构思，并且想在历史数据中看看它的表现的话，PyAlgoTrade应该能够给你一些帮助。

感谢要放在最开始——感谢豪尔赫帮助审查初步设计和文档。

本教程是在UNIX下开发的，但是把它调整为windows环境应该是比较简单的。 因为我就是在windows下翻译的。

PyAlgoTrade有6大主要组件:

- Strategies策略
- Feeds数据源
- Brokers经纪商
- DataSeries数据序列
- Technicals指标计算
- Optimizer优化

Strategies 这是你定义的实现交易逻辑的类：何时买、何时卖，等等。

Feeds These are data providing abstractions. 例如，你可以使用CSV数据源从一个格式化后的csv(以逗号分割)文件中加载数据推送给策略。

数据源不仅限于bars。例如，可以利用Twitter数据源将Twitter的事件信息转化为交易决策（译者注：事件驱动）。

Brokers 经纪商模块负责执行订单。

DataSeries DataSeries 是用于管理时间序列的抽象类

Technicals 这是你用来对DataSeries进行计算的一组过滤（指标）器。 例如简单移动平均线（SMA）,相对强弱指标(RSI)等. 这些过滤(指标)器被建模为DataSeries 的装饰器。

Optimizer 这是能让你在不同电脑之间、或多进程、或二者结合以加快回测效率的一组类。

说到这里，我们测试策略时需要一些数据，让我们使用以下命令获取甲骨文(Oracle)2000年的数据:

```
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('orcl', 2000, 'orcl-2000.csv')"
```

pyalgotrade.tools.yahoofinance 从雅虎金融(Yahoo! Finance)上打包下载格式化后的CSV数据。 文件orcl-2000.csv的格式和内容如下:

```
Date,Open,High,Low,Close,Volume,Adj Close
2000-12-29,30.87,31.31,28.69,29.06,31655500,28.35
2000-12-28,30.56,31.12,30.37,31.06,25055600,30.30
2000-12-27,30.37,31.06,29.37,30.69,26441700,29.94
.
.
2000-01-04,115.50,118.62,105.00,107.69,116850000,26.26
2000-01-03,124.62,125.19,111.62,118.12,98122000,28.81
```

让我们从一个简单的策略开始，即在运行过程中只打印收盘价:

```
from pyalgotrade import strategy
from pyalgotrade.barfeed import yahoofeed


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument

    def onBars(self, bars):
        bar = bars[self.__instrument]
        self.info(bar.getClose())

# Load the yahoo feed from the CSV file
feed = yahoofeed.Feed()
feed.addBarsFromCSV("orcl", "orcl-2000.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = MyStrategy(feed, "orcl")
myStrategy.run()
```

这段代码主要做了3件事情:(???)

1. 声明了一个新的策略，这个策略中只定义了一个方法——*onBars*,这个方法会在推送的每一个bar执行(???)。
2. 从一个CSV文件载入数据源
3. 在数据源推送的每根bar上执行策略。

如果你运行了这个脚本，你应该在命令中看到收盘价:

```
2000-01-03 00:00:00 strategy [INFO] 118.12
2000-01-04 00:00:00 strategy [INFO] 107.69
2000-01-05 00:00:00 strategy [INFO] 102.0
.
.
.
2000-12-27 00:00:00 strategy [INFO] 30.69
2000-12-28 00:00:00 strategy [INFO] 31.06
2000-12-29 00:00:00 strategy [INFO] 29.06
```

下一步我们实现一个打印SMA价格的策略，用来演示technicals是如何被使用的：

```
from pyalgotrade import strategy
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.technical import ma


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument):
        strategy.BacktestingStrategy.__init__(self, feed)
        # We want a 15 period SMA over the closing prices.
        self.__sma = ma.SMA(feed[instrument].getCloseDataSeries(), 15)
        self.__instrument = instrument

    def onBars(self, bars):
        bar = bars[self.__instrument]
        self.info("%s %s" % (bar.getClose(), self.__sma[-1]))

# Load the yahoo feed from the CSV file
feed = yahoofeed.Feed()
feed.addBarsFromCSV("orcl", "orcl-2000.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = MyStrategy(feed, "orcl")
myStrategy.run()
```

这跟之前的例子很相似，除了：

1、我们在收盘价基础上初始化了一个SMA过滤器（指标）。

2、我们将SMA的值随着收盘价一起打印出来。

运行该脚本后将会看到逐行对应的收盘价和SMA值，但是实际情况是前14行的SMA值是None。

这是因为我们至少需要15个值才能把SMA计算出来：

```
2000-01-03 00:00:00 strategy [INFO] 118.12 None
2000-01-04 00:00:00 strategy [INFO] 107.69 None
2000-01-05 00:00:00 strategy [INFO] 102.0 None
2000-01-06 00:00:00 strategy [INFO] 96.0 None
2000-01-07 00:00:00 strategy [INFO] 103.37 None
2000-01-10 00:00:00 strategy [INFO] 115.75 None
2000-01-11 00:00:00 strategy [INFO] 112.37 None
2000-01-12 00:00:00 strategy [INFO] 105.62 None
2000-01-13 00:00:00 strategy [INFO] 105.06 None
2000-01-14 00:00:00 strategy [INFO] 106.81 None
2000-01-18 00:00:00 strategy [INFO] 111.25 None
2000-01-19 00:00:00 strategy [INFO] 57.13 None
2000-01-20 00:00:00 strategy [INFO] 59.25 None
2000-01-21 00:00:00 strategy [INFO] 59.69 None
2000-01-24 00:00:00 strategy [INFO] 54.19 94.2866666667
2000-01-25 00:00:00 strategy [INFO] 56.44 90.1746666667
.
.
.
2000-12-27 00:00:00 strategy [INFO] 30.69 29.9866666667
2000-12-28 00:00:00 strategy [INFO] 31.06 30.0446666667
2000-12-29 00:00:00 strategy [INFO] 29.06 30.0946666667
```

若给定的时间不足以计算，则所有的technicals将返回None。

一件重要的事情是technicals可以进行组合，因为他们作为DataSeries是模块化的。

比如，计算收盘价的RSI再计算RSI的SMA就很简单：

```
from pyalgotrade import strategy
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.technical import ma
from pyalgotrade.technical import rsi


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__rsi = rsi.RSI(feed[instrument].getCloseDataSeries(), 14)
        self.__sma = ma.SMA(self.__rsi, 15)
        self.__instrument = instrument

    def onBars(self, bars):
        bar = bars[self.__instrument]
        self.info("%s %s %s" % (bar.getClose(), self.__rsi[-1], self.__sma[-1]))

# Load the yahoo feed from the CSV file
feed = yahoofeed.Feed()
feed.addBarsFromCSV("orcl", "orcl-2000.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = MyStrategy(feed, "orcl")
myStrategy.run()
```

运行该脚本，你应该看到在屏幕上有一堆值：

- 前14个RSI值是None。因为我们至少需要15个值来计算RSI。
- 前28个SMA值是None。前14个RSI值是None，SMA过滤器（指标）接收的数据从第15个开始才不是None。我们计算SMA(15)至少需要15个非None值。

```
2000-01-03 00:00:00 strategy [INFO] 118.12 None None
2000-01-04 00:00:00 strategy [INFO] 107.69 None None
2000-01-05 00:00:00 strategy [INFO] 102.0 None None
2000-01-06 00:00:00 strategy [INFO] 96.0 None None
2000-01-07 00:00:00 strategy [INFO] 103.37 None None
2000-01-10 00:00:00 strategy [INFO] 115.75 None None
2000-01-11 00:00:00 strategy [INFO] 112.37 None None
2000-01-12 00:00:00 strategy [INFO] 105.62 None None
2000-01-13 00:00:00 strategy [INFO] 105.06 None None
2000-01-14 00:00:00 strategy [INFO] 106.81 None None
2000-01-18 00:00:00 strategy [INFO] 111.25 None None
2000-01-19 00:00:00 strategy [INFO] 57.13 None None
2000-01-20 00:00:00 strategy [INFO] 59.25 None None
2000-01-21 00:00:00 strategy [INFO] 59.69 None None
2000-01-24 00:00:00 strategy [INFO] 54.19 23.5673530141 None
2000-01-25 00:00:00 strategy [INFO] 56.44 25.0687519877 None
2000-01-26 00:00:00 strategy [INFO] 55.06 24.7476577095 None
2000-01-27 00:00:00 strategy [INFO] 51.81 23.9690136517 None
2000-01-28 00:00:00 strategy [INFO] 47.38 22.9108539956 None
2000-01-31 00:00:00 strategy [INFO] 49.95 24.980004823 None
2000-02-01 00:00:00 strategy [INFO] 54.0 28.2484181864 None
2000-02-02 00:00:00 strategy [INFO] 54.31 28.505177315 None
2000-02-03 00:00:00 strategy [INFO] 56.69 30.5596770599 None
2000-02-04 00:00:00 strategy [INFO] 57.81 31.5564353751 None
```

```
2000-02-07 00:00:00 strategy [INFO] 59.94 33.5111056589 None
2000-02-08 00:00:00 strategy [INFO] 59.56 33.3282358994 None
2000-02-09 00:00:00 strategy [INFO] 59.94 33.7177605915 None
2000-02-10 00:00:00 strategy [INFO] 62.31 36.2205441255 None
2000-02-11 00:00:00 strategy [INFO] 59.69 34.6623493641 29.0368892505
2000-02-14 00:00:00 strategy [INFO] 62.19 37.4284445543 29.9609620198
.
.
.
2000-12-27 00:00:00 strategy [INFO] 30.69 51.3196802735 49.8506368511
2000-12-28 00:00:00 strategy [INFO] 31.06 52.1646203455 49.997518354
2000-12-29 00:00:00 strategy [INFO] 29.06 47.3776678335 50.0790646925
```

交易

现在让我们用一个简单的策略模拟实际的交易，思路很简单：

- 如果复权价格在SMA(15)之上时，我们买入一个多单(我们发出一张买订单).
- 如果我们持有一张多单，当复权价跌破SMA(15)时，我们平多单(我们发出一张卖单).

```python
from pyalgotrade import strategy
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.technical import ma


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, smaPeriod):
        strategy.BacktestingStrategy.__init__(self, feed, 1000)
        self.__position = None
        self.__instrument = instrument
        # We'll use adjusted close values instead of regular close values.
        self.setUseAdjustedValues(True)
        self.__sma = ma.SMA(feed[instrument].getPriceDataSeries(), smaPeriod)

    def onEnterOk(self, position):
        execInfo = position.getEntryOrder().getExecutionInfo()
        self.info("BUY at $%.2f" % (execInfo.getPrice()))

    def onEnterCanceled(self, position):
        self.__position = None

    def onExitOk(self, position):
        execInfo = position.getExitOrder().getExecutionInfo()
        self.info("SELL at $%.2f" % (execInfo.getPrice()))
        self.__position = None

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        self.__position.exitMarket()

    def onBars(self, bars):
        # Wait for enough bars to be available to calculate a SMA.
        if self.__sma[-1] is None:
            return
        bar = bars[self.__instrument]
        # If a position was not opened, check if we should enter a long position.
        if self.__position is None:
            if bar.getPrice() > self.__sma[-1]:
                # Enter a buy market order for 10 shares. The order is good till␣
↪canceled.
                self.__position = self.enterLong(self.__instrument, 10, True)
        # Check if we have to exit the position.
        elif bar.getPrice() < self.__sma[-1] and not self.__position.exitActive():
            self.__position.exitMarket()


def run_strategy(smaPeriod):
    # Load the yahoo feed from the CSV file
    feed = yahoofeed.Feed()
    feed.addBarsFromCSV("orcl", "orcl-2000.csv")

    # Evaluate the strategy with the feed.
    myStrategy = MyStrategy(feed, "orcl", smaPeriod)
    myStrategy.run()
    print(("Final portfolio value: $%.2f" % myStrategy.getBroker().getEquity()))

run_strategy(15)
```

运行该脚本，你将看到跟下面类似的东西:

```
2000-01-26 00:00:00 strategy [INFO] BUY at $27.26
2000-01-28 00:00:00 strategy [INFO] SELL at $24.74
2000-02-03 00:00:00 strategy [INFO] BUY at $26.60
2000-02-22 00:00:00 strategy [INFO] SELL at $28.40
2000-02-23 00:00:00 strategy [INFO] BUY at $28.91
2000-03-31 00:00:00 strategy [INFO] SELL at $38.51
2000-04-07 00:00:00 strategy [INFO] BUY at $40.19
2000-04-12 00:00:00 strategy [INFO] SELL at $37.44
2000-04-19 00:00:00 strategy [INFO] BUY at $37.76
2000-04-20 00:00:00 strategy [INFO] SELL at $35.45
2000-04-28 00:00:00 strategy [INFO] BUY at $37.70
2000-05-05 00:00:00 strategy [INFO] SELL at $35.54
2000-05-08 00:00:00 strategy [INFO] BUY at $36.17
2000-05-09 00:00:00 strategy [INFO] SELL at $35.39
2000-05-16 00:00:00 strategy [INFO] BUY at $37.28
2000-05-19 00:00:00 strategy [INFO] SELL at $34.58
2000-05-31 00:00:00 strategy [INFO] BUY at $35.18
2000-06-23 00:00:00 strategy [INFO] SELL at $38.81
2000-06-27 00:00:00 strategy [INFO] BUY at $39.56
2000-06-28 00:00:00 strategy [INFO] SELL at $39.42
2000-06-29 00:00:00 strategy [INFO] BUY at $39.41
2000-06-30 00:00:00 strategy [INFO] SELL at $38.60
2000-07-03 00:00:00 strategy [INFO] BUY at $38.96
2000-07-05 00:00:00 strategy [INFO] SELL at $36.89
2000-07-21 00:00:00 strategy [INFO] BUY at $37.19
2000-07-24 00:00:00 strategy [INFO] SELL at $37.04
2000-07-26 00:00:00 strategy [INFO] BUY at $35.93
2000-07-28 00:00:00 strategy [INFO] SELL at $36.08
2000-08-01 00:00:00 strategy [INFO] BUY at $36.11
2000-08-02 00:00:00 strategy [INFO] SELL at $35.06
2000-08-04 00:00:00 strategy [INFO] BUY at $37.61
2000-09-11 00:00:00 strategy [INFO] SELL at $41.34
```

```
2000-09-29 00:00:00 strategy [INFO] BUY at $39.07
2000-10-02 00:00:00 strategy [INFO] SELL at $38.30
2000-10-20 00:00:00 strategy [INFO] BUY at $34.71
2000-10-31 00:00:00 strategy [INFO] SELL at $31.34
2000-11-20 00:00:00 strategy [INFO] BUY at $23.35
2000-11-21 00:00:00 strategy [INFO] SELL at $23.83
2000-12-01 00:00:00 strategy [INFO] BUY at $25.33
2000-12-21 00:00:00 strategy [INFO] SELL at $26.72
2000-12-22 00:00:00 strategy [INFO] BUY at $29.17
Final portfolio value: $979.44
```

如果我们用30替代15来作为SMA的参数会怎样？绩效会变好还是变坏？ 我们很确定可以通过以下方法来验证：

```python
for i in range(10, 30):
    run_strategy(i)
```

我们会发现，使用SMA(20)后能得到更好的回报：

```
Final portfolio value: $1075.38
```

我们只是尝试了一组有限的参数组，看起来没有什么问题。如果我们需要测试一个有多个参数的策略怎么办？

使用串行计算就不会让策略变得更复杂。

优化

了解优化组件后，优化会变得非常简单：

- **有一个服务负责：**
    - 提供策略运行所需的bars。
    - 提供策略运行所需的参数。
    - 记录来自每个工作过程的交易记录结果。
- **并且有多个工作过程负责：**
    - 在由服务提供的参数、K线基础上运行策略。

我们使用一个名为RSI2的策略来说明上述架构(http://stockcharts.com/school/doku.php?id=chart_school:trading_strategies:rsi2), 它需要以下参数：

- entrySMA：一个SMA趋势识别的长度。参数范围[150 ~ 250]。
- exitSMA：一个较短的SMA出场点长度。参数范围[5~10]。
- rsiPeriod：多空入场点。参数范围[2 ~ 10]。
- overSoldThreshold：RSI多单平仓点。参数范围[5 ~ 25]。
- overBoughtThreshold：RSI空单平仓点。参数范围[75 ~ 95]。

如果我没算错的话——你的意思是数学是体育老师教的？？————这将有4409559个参数组合。

我这里测试其中一个参数组的时间大约0.16秒，估计需要8.5天才能要运行完整个参数组合然后找到最优参数。时间太长了。如果我能有10台8核的机器同时进行优化，那么时间将缩短到2.5小时。

长话短说，**我们需要并行计算.**

我们先下载道琼斯工业指数3年的日线数据:

```
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('dia', 2009, 'dia-2009.csv')"
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('dia', 2010, 'dia-2010.csv')"
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('dia', 2011, 'dia-2011.csv')"
```

保存以下代码到rsi2.py:

```
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import rsi
from pyalgotrade.technical import cross


class RSI2(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, entrySMA, exitSMA, rsiPeriod,␣
↪overBoughtThreshold, overSoldThreshold):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        # We'll use adjusted close values, if available, instead of regular close␣
↪values.
        if feed.barsHaveAdjClose():
            self.setUseAdjustedValues(True)
        self.__priceDS = feed[instrument].getPriceDataSeries()
        self.__entrySMA = ma.SMA(self.__priceDS, entrySMA)
        self.__exitSMA = ma.SMA(self.__priceDS, exitSMA)
        self.__rsi = rsi.RSI(self.__priceDS, rsiPeriod)
        self.__overBoughtThreshold = overBoughtThreshold
        self.__overSoldThreshold = overSoldThreshold
        self.__longPos = None
        self.__shortPos = None

    def getEntrySMA(self):
        return self.__entrySMA

    def getExitSMA(self):
        return self.__exitSMA

    def getRSI(self):
        return self.__rsi

    def onEnterCanceled(self, position):
        if self.__longPos == position:
            self.__longPos = None
        elif self.__shortPos == position:
            self.__shortPos = None
        else:
            assert(False)

    def onExitOk(self, position):
        if self.__longPos == position:
            self.__longPos = None
        elif self.__shortPos == position:
            self.__shortPos = None
        else:
            assert(False)
    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        position.exitMarket()

    def onBars(self, bars):
        # Wait for enough bars to be available to calculate SMA and RSI.
        if self.__exitSMA[-1] is None or self.__entrySMA[-1] is None or self.__rsi[-
↪1] is None:
            return

        bar = bars[self.__instrument]
        if self.__longPos is not None:
            if self.exitLongSignal():
                self.__longPos.exitMarket()
        elif self.__shortPos is not None:
            if self.exitShortSignal():
                self.__shortPos.exitMarket()
        else:
            if self.enterLongSignal(bar):
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                self.__longPos = self.enterLong(self.__instrument, shares, True)
            elif self.enterShortSignal(bar):
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                self.__shortPos = self.enterShort(self.__instrument, shares, True)

    def enterLongSignal(self, bar):
        return bar.getPrice() > self.__entrySMA[-1] and self.__rsi[-1] <= self.__
↪overSoldThreshold

    def exitLongSignal(self):
        return cross.cross_above(self.__priceDS, self.__exitSMA) and not self.__
↪longPos.exitActive()

    def enterShortSignal(self, bar):
        return bar.getPrice() < self.__entrySMA[-1] and self.__rsi[-1] >= self.__
↪overBoughtThreshold

    def exitShortSignal(self):
        return cross.cross_below(self.__priceDS, self.__exitSMA) and not self.__
↪shortPos.exitActive()
```

这是服务器脚本:

```
import itertools
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.optimizer import server


def parameters_generator():
    instrument = ["dia"]
    entrySMA = list(range(150, 251))
    exitSMA = list(range(5, 16))
    rsiPeriod = list(range(2, 11))
    overBoughtThreshold = list(range(75, 96))
    overSoldThreshold = list(range(5, 26))
    return itertools.product(instrument, entrySMA, exitSMA, rsiPeriod,␣
↪overBoughtThreshold, overSoldThreshold)

# The if __name__ == '__main__' part is necessary if running on Windows.
if __name__ == '__main__':
    # Load the feed from the CSV files.
    feed = yahoofeed.Feed()
    feed.addBarsFromCSV("dia", "dia-2009.csv")
    feed.addBarsFromCSV("dia", "dia-2010.csv")
    feed.addBarsFromCSV("dia", "dia-2011.csv")

    # Run the server.
    server.serve(feed, parameters_generator(), "localhost", 5000)
```

服务器脚本做了3件事:

1. 声明一个生成器函数，这个函数可以根据参数范围为策略生成参数组合。
2. 从我们下载的CSV文件载入数据源。
3. 运行服务端并在端口5000上等待连接。

This is the worker script that uses the **pyalgotrade.optimizer.worker** module to run the strategy in parallel with the data supplied by the server:

```
from pyalgotrade.optimizer import worker
import rsi2

# The if __name__ == '__main__' part is necessary if running on Windows.
if __name__ == '__main__':
    worker.run(rsi2.RSI2, "localhost", 5000, workerName="localworker")
```

当您运行服务器和客户端时，您会看到在服务器控制台上看到的类似下面的内容：

```
2014-05-03 15:04:01,083 server [INFO] Loading bars
2014-05-03 15:04:01,348 server [INFO] Waiting for workers
2014-05-03 15:04:58,277 server [INFO] Partial result 1242173.28754 with parameters: (
↪'dia', 150, 5, 2, 91, 19) from localworker
2014-05-03 15:04:58,566 server [INFO] Partial result 1203266.33502 with parameters: (
↪'dia', 150, 5, 2, 81, 19) from localworker
2014-05-03 15:05:50,965 server [INFO] Partial result 1220763.1579 with parameters: (
↪'dia', 150, 5, 3, 83, 24) from localworker
2014-05-03 15:05:51,325 server [INFO] Partial result 1221627.50793 with parameters: (
↪'dia', 150, 5, 3, 80, 24) from localworker
.
.
```

在客户机控制台可以看到下面的内容:

```
2014-05-03 15:02:25,360 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 84, 15)
2014-05-03 15:02:25,377 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 94, 5)
2014-05-03 15:02:25,661 localworker [INFO] Result 1090481.06342
2014-05-03 15:02:25,661 localworker [INFO] Result 1031470.23717
2014-05-03 15:02:25,662 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 93, 25)
2014-05-03 15:02:25,665 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 84, 14)
```

```
2014-05-03 15:02:25,995 localworker [INFO] Result 1135558.55667
2014-05-03 15:02:25,996 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 93, 24)
2014-05-03 15:02:26,006 localworker [INFO] Result 1083987.18174
2014-05-03 15:02:26,007 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 84, 13)
2014-05-03 15:02:26,256 localworker [INFO] Result 1093736.17175
2014-05-03 15:02:26,257 localworker [INFO] Running strategy with parameters ('dia',␣
↪150, 5, 2, 84, 12)
2014-05-03 15:02:26,280 localworker [INFO] Result 1135558.55667
.
.
```

注意！！**你只能运行一个服务器和一个或者多个客户机。**。译者补充：这里说的是同一个优化项目下只能运行一个服务器。

如果你只想在自己的电脑上对策略并行计算，你可以利用 **pyalgotrade.optimizer.local** 模块来实现：

```
import itertools
from pyalgotrade.optimizer import local
from pyalgotrade.barfeed import yahoofeed
import rsi2


def parameters_generator():
    instrument = ["dia"]
    entrySMA = list(range(150, 251))
    exitSMA = list(range(5, 16))
    rsiPeriod = list(range(2, 11))
    overBoughtThreshold = list(range(75, 96))
    overSoldThreshold = list(range(5, 26))
    return itertools.product(instrument, entrySMA, exitSMA, rsiPeriod,␣
↪overBoughtThreshold, overSoldThreshold)


# The if __name__ == '__main__' part is necessary if running on Windows.
if __name__ == '__main__':
    # Load the feed from the CSV files.
    feed = yahoofeed.Feed()
    feed.addBarsFromCSV("dia", "dia-2009.csv")
    feed.addBarsFromCSV("dia", "dia-2010.csv")
    feed.addBarsFromCSV("dia", "dia-2011.csv")

    local.run(rsi2.RSI2, feed, parameters_generator())
```

上述代码做了3件事:

1. 声明一个生成器函数，这个函数可以根据参数范围为策略生成参数组合。
2. 从我们下载的CSV文件载入数据源。
3. 通过 **pyalgotrade.optimizer.local** 模块运行策略并寻优。

当你执行这段代码后，你可以看到类似下面的结果:

```
2014-05-03 15:08:06,587 server [INFO] Loading bars
2014-05-03 15:08:06,910 server [INFO] Waiting for workers
2014-05-03 15:08:58,347 server [INFO] Partial result 1242173.28754 with parameters: (
↪'dia', 150, 5, 2, 91, 19) from worker-95583
2014-05-03 15:08:58,967 server [INFO] Partial result 1203266.33502 with parameters: (
↪'dia', 150, 5, 2, 81, 19) from worker-95584
```

```
2014-05-03 15:09:52,097 server [INFO] Partial result 1220763.1579 with parameters: (
↪'dia', 150, 5, 3, 83, 24) from worker-95584
2014-05-03 15:09:52,921 server [INFO] Partial result 1221627.50793 with parameters: (
↪'dia', 150, 5, 3, 80, 24) from worker-95583
2014-05-03 15:10:40,826 server [INFO] Partial result 1142162.23912 with parameters: (
↪'dia', 150, 5, 4, 76, 17) from worker-95584
2014-05-03 15:10:41,318 server [INFO] Partial result 1107487.03214 with parameters: (
↪'dia', 150, 5, 4, 83, 17) from worker-95583
.
.
```

在记录中，最优结果为 **$2314.40** ,它的参数组是:

1. entrySMA: 154
2. exitSMA: 5
3. rsiPeriod: 2
4. overBoughtThreshold: 91
5. overSoldThreshold: 18

绘图

PyAlgoTrade可以很容易的对策略运行进行绘图。

保存为 sma_crossover.py:

```
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import cross


class SMACrossOver(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, smaPeriod):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        self.__position = None
        # We'll use adjusted close values instead of regular close values.
        self.setUseAdjustedValues(True)
        self.__prices = feed[instrument].getPriceDataSeries()
        self.__sma = ma.SMA(self.__prices, smaPeriod)

    def getSMA(self):
        return self.__sma

    def onEnterCanceled(self, position):
        self.__position = None

    def onExitOk(self, position):
        self.__position = None

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        self.__position.exitMarket()

    def onBars(self, bars):
        # If a position was not opened, check if we should enter a long position.
        if self.__position is None:
            if cross.cross_above(self.__prices, self.__sma) > 0:
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                # Enter a buy market order. The order is good till canceled.
                self.__position = self.enterLong(self.__instrument, shares, True)
        # Check if we have to exit the position.
        elif not self.__position.exitActive() and cross.cross_below(self.__prices,␣
↪self.__sma) > 0:
            self.__position.exitMarket()
```

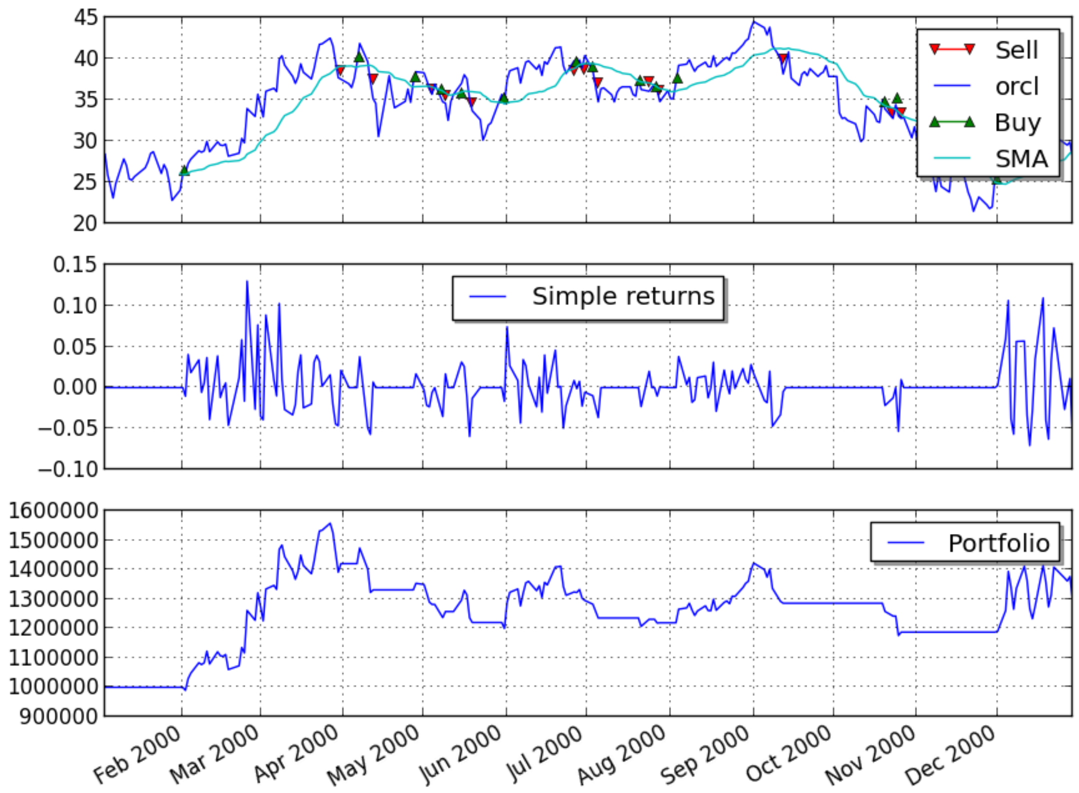
保存这段代码到另外一个文件:

```
from pyalgotrade import plotter
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.stratanalyzer import returns
import sma_crossover

# Load the yahoo feed from the CSV file
feed = yahoofeed.Feed()
feed.addBarsFromCSV("orcl", "orcl-2000.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = sma_crossover.SMACrossOver(feed, "orcl", 20)

# Attach a returns analyzers to the strategy.
returnsAnalyzer = returns.Returns()
myStrategy.attachAnalyzer(returnsAnalyzer)

# Attach the plotter to the strategy.
plt = plotter.StrategyPlotter(myStrategy)
# Include the SMA in the instrument's subplot to get it displayed along with the␣
↪closing prices.
plt.getInstrumentSubplot("orcl").addDataSeries("SMA", myStrategy.getSMA())
# Plot the simple returns on each bar.
plt.getOrCreateSubplot("returns").addDataSeries("Simple returns", returnsAnalyzer.
↪getReturns())

# Run the strategy.
myStrategy.run()
myStrategy.info("Final portfolio value: $%.2f" % myStrategy.getResult())

# Plot the strategy.
plt.plot()
```

上述代码做了3件事:

1. 从一个CSV文件中加载数据源。
2. 通过数据源提供的bar逐根运行策略；提供一个策略绘图程序。
3. 绘制策略图形。

图形如下:



我希望你能喜欢这份简明教程. 同时提醒你下载地址在这里: http://gbeced.github.io/pyalgotrade/downloads/index.html

开始编写你自己的策略吧!

你可以在 策略例子 找到更多的例子.

CHAPTER 3

代码的文档说明

目录:

bar – 品种的价格

dataseries – 数据序列基类

数据序列是管理time-series数据的抽象类。

feed – 数据源基类

Feeds 是将时间序列抽象化 当这些在事件调度循环时，会推送一个新的可用数据。 When these are included in the event dispatch loop, they emit an event as new data is available. Feeds are also responsible for updating the `pyalgotrade.dataseries.DataSeries` associated with each piece of data that the feed provides.

当前位置是关于数据源基类的，要了解bar的数据源请查阅： *barfeed – Bar*数据来源 **板块.**

CSV 支持

CSV 支持的例子

一个带有以下格式的文件

```
Date,USD,GBP,EUR
2013-09-29,1333.0,831.203,986.75
2013-09-22,1349.25,842.755,997.671
2013-09-15,1318.5,831.546,993.969
2013-09-08,1387.0,886.885,1052.911
```

```
.
.
.
```

像这样引入并使用:

```
from pyalgotrade.feed import csvfeed

feed = csvfeed.Feed("Date", "%Y-%m-%d")
feed.addValuesFromCSV("quandl_gold_2.csv")
for dateTime, value in feed:
    print(dateTime, value)
```

输出结果如下:

```
1968-04-07 00:00:00 {'USD': 37.0, 'GBP': 15.3875, 'EUR': ''}
1968-04-14 00:00:00 {'USD': 38.0, 'GBP': 15.8208, 'EUR': ''}
1968-04-21 00:00:00 {'USD': 37.65, 'GBP': 15.6833, 'EUR': ''}
1968-04-28 00:00:00 {'USD': 38.65, 'GBP': 16.1271, 'EUR': ''}
1968-05-05 00:00:00 {'USD': 39.1, 'GBP': 16.3188, 'EUR': ''}
1968-05-12 00:00:00 {'USD': 39.6, 'GBP': 16.5625, 'EUR': ''}
1968-05-19 00:00:00 {'USD': 41.5, 'GBP': 17.3958, 'EUR': ''}
1968-05-26 00:00:00 {'USD': 41.75, 'GBP': 17.5104, 'EUR': ''}
1968-06-02 00:00:00 {'USD': 41.95, 'GBP': 17.6, 'EUR': ''}
1968-06-09 00:00:00 {'USD': 41.25, 'GBP': 17.3042, 'EUR': ''}
.
.
.
2013-07-28 00:00:00 {'USD': 1331.0, 'GBP': 864.23, 'EUR': 1001.505}
2013-08-04 00:00:00 {'USD': 1309.25, 'GBP': 858.637, 'EUR': 986.921}
2013-08-11 00:00:00 {'USD': 1309.0, 'GBP': 843.156, 'EUR': 979.79}
2013-08-18 00:00:00 {'USD': 1369.25, 'GBP': 875.424, 'EUR': 1024.964}
2013-08-25 00:00:00 {'USD': 1377.5, 'GBP': 885.738, 'EUR': 1030.6}
2013-09-01 00:00:00 {'USD': 1394.75, 'GBP': 901.292, 'EUR': 1055.749}
2013-09-08 00:00:00 {'USD': 1387.0, 'GBP': 886.885, 'EUR': 1052.911}
2013-09-15 00:00:00 {'USD': 1318.5, 'GBP': 831.546, 'EUR': 993.969}
2013-09-22 00:00:00 {'USD': 1349.25, 'GBP': 842.755, 'EUR': 997.671}
2013-09-29 00:00:00 {'USD': 1333.0, 'GBP': 831.203, 'EUR': 986.75}
```

barfeed – Bar数据来源

CSV

Yahoo! Finance

Google Finance

Quandl

Ninja Trader

technical – 技术指标

例子

下面的这个例子演示了如何结合 `EventWindow` 和 `EventBasedFilter` 来计算一个指标:

```python
from pyalgotrade import dataseries
from pyalgotrade import technical


# An EventWindow is responsible for making calculations using a window of values.
class Accumulator(technical.EventWindow):
    def getValue(self):
        ret = None
        if self.windowFull():
            ret = self.getValues().sum()
        return ret

# Build a sequence based DataSeries.
seqDS = dataseries.SequenceDataSeries()
# Wrap it with a filter that will get fed as new values get added to the underlying␣
↪DataSeries.
accum = technical.EventBasedFilter(seqDS, Accumulator(3))

# Put in some values.
for i in range(0, 50):
    seqDS.append(i)

# Get some values.
print(accum[0])  # Not enough values yet.
print(accum[1])  # Not enough values yet.
print(accum[2])  # Ok, now we should have at least 3 values.
print(accum[3])

# Get the last value, which should be equal to 49 + 48 + 47.
print(accum[-1])
```

输出内容如下:

```
None
None
3.0
```

```
6.0
144.0
```

移动平均线

动量指标

其他指标

broker – 订单管理类

基类和模块

回测模块和类

strategy – 策略基类

Strategies类是你定义的何时买、何时卖等等交易逻辑的类。

做多和做空可以通过以下两种方式:

- 使用下面任意一个方法下单:
- `pyalgotrade.strategy.BaseStrategy.marketOrder()`
- `pyalgotrade.strategy.BaseStrategy.limitOrder()`
- `pyalgotrade.strategy.BaseStrategy.stopOrder()`
- `pyalgotrade.strategy.BaseStrategy.stopLimitOrder()`
- 使用封装了买/卖的高级接口下单:
- `pyalgotrade.strategy.BaseStrategy.enterLong()`
- `pyalgotrade.strategy.BaseStrategy.enterShort()`
- `pyalgotrade.strategy.BaseStrategy.enterLongLimit()`
- `pyalgotrade.strategy.BaseStrategy.enterShortLimit()`

Positions are higher level abstractions for placing orders. They are escentially a pair of entry-exit orders and provide easier tracking for returns and PnL than using individual orders.

Strategy

Position

stratanalyzer – Strategy analyzers

Strategy analyzers provide an extensible way to attach different calculations to strategy executions.

class `pyalgotrade.stratanalyzer.`**`StrategyAnalyzer`**

Bases: `object`

Base class for strategy analyzers.

注解： This is a base class and should not be used directly.

Returns

Sharpe Ratio

DrawDown

class `pyalgotrade.stratanalyzer.drawdown.`**`DrawDown`**

Bases: *`pyalgotrade.stratanalyzer.StrategyAnalyzer`*

A *`pyalgotrade.stratanalyzer.StrategyAnalyzer`* that calculates max. drawdown and longest drawdown duration for the portfolio.

`getLongestDrawDownDuration``()`

Returns the duration of the longest drawdown.

返回类型 `datetime.timedelta`.

注解： Note that this is the duration of the longest drawdown, not necessarily the deepest one.

`getMaxDrawDown``()`

Returns the max. (deepest) drawdown.

Trades

Example

Save this code as sma_crossover.py:

```
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import cross


class SMACrossOver(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, smaPeriod):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        self.__position = None
        # We'll use adjusted close values instead of regular close values.
        self.setUseAdjustedValues(True)
        self.__prices = feed[instrument].getPriceDataSeries()
        self.__sma = ma.SMA(self.__prices, smaPeriod)

    def getSMA(self):
        return self.__sma
    def onEnterCanceled(self, position):
        self.__position = None

    def onExitOk(self, position):
        self.__position = None

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        self.__position.exitMarket()

    def onBars(self, bars):
        # If a position was not opened, check if we should enter a long position.
        if self.__position is None:
            if cross.cross_above(self.__prices, self.__sma) > 0:
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                # Enter a buy market order. The order is good till canceled.
                self.__position = self.enterLong(self.__instrument, shares, True)
        # Check if we have to exit the position.
        elif not self.__position.exitActive() and cross.cross_below(self.__prices,␣
↪self.__sma) > 0:
            self.__position.exitMarket()
```

and save this code in a different file:

```
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.stratanalyzer import returns
from pyalgotrade.stratanalyzer import sharpe
from pyalgotrade.stratanalyzer import drawdown
from pyalgotrade.stratanalyzer import trades
import sma_crossover

# Load the yahoo feed from the CSV file
feed = yahoofeed.Feed()
feed.addBarsFromCSV("orcl", "orcl-2000.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = sma_crossover.SMACrossOver(feed, "orcl", 20)

# Attach different analyzers to a strategy before executing it.
retAnalyzer = returns.Returns()
myStrategy.attachAnalyzer(retAnalyzer)
sharpeRatioAnalyzer = sharpe.SharpeRatio()
myStrategy.attachAnalyzer(sharpeRatioAnalyzer)
drawDownAnalyzer = drawdown.DrawDown()
myStrategy.attachAnalyzer(drawDownAnalyzer)
tradesAnalyzer = trades.Trades()
myStrategy.attachAnalyzer(tradesAnalyzer)

# Run the strategy.
myStrategy.run()

print("Final portfolio value: $%.2f" % myStrategy.getResult())
print("Cumulative returns: %.2f %%" % (retAnalyzer.getCumulativeReturns()[-1] * 100))
print("Sharpe ratio: %.2f" % (sharpeRatioAnalyzer.getSharpeRatio(0.05)))
print("Max. drawdown: %.2f %%" % (drawDownAnalyzer.getMaxDrawDown() * 100))
print("Longest drawdown duration: %s" % (drawDownAnalyzer.
↪getLongestDrawDownDuration()))
```

```
print()
print("Total trades: %d" % (tradesAnalyzer.getCount()))
if tradesAnalyzer.getCount() > 0:
    profits = tradesAnalyzer.getAll()
    print("Avg. profit: $%2.f" % (profits.mean()))
    print("Profits std. dev.: $%2.f" % (profits.std()))
    print("Max. profit: $%2.f" % (profits.max()))
    print("Min. profit: $%2.f" % (profits.min()))
    returns = tradesAnalyzer.getAllReturns()
    print("Avg. return: %2.f %%" % (returns.mean() * 100))
    print("Returns std. dev.: %2.f %%" % (returns.std() * 100))
    print("Max. return: %2.f %%" % (returns.max() * 100))
    print("Min. return: %2.f %%" % (returns.min() * 100))

print()
print("Profitable trades: %d" % (tradesAnalyzer.getProfitableCount()))
if tradesAnalyzer.getProfitableCount() > 0:
    profits = tradesAnalyzer.getProfits()
    print("Avg. profit: $%2.f" % (profits.mean()))
    print("Profits std. dev.: $%2.f" % (profits.std()))
    print("Max. profit: $%2.f" % (profits.max()))
    print("Min. profit: $%2.f" % (profits.min()))
    returns = tradesAnalyzer.getPositiveReturns()
    print("Avg. return: %2.f %%" % (returns.mean() * 100))
    print("Returns std. dev.: %2.f %%" % (returns.std() * 100))
    print("Max. return: %2.f %%" % (returns.max() * 100))
    print("Min. return: %2.f %%" % (returns.min() * 100))

print()
print("Unprofitable trades: %d" % (tradesAnalyzer.getUnprofitableCount()))
if tradesAnalyzer.getUnprofitableCount() > 0:
    losses = tradesAnalyzer.getLosses()
    print("Avg. loss: $%2.f" % (losses.mean()))
    print("Losses std. dev.: $%2.f" % (losses.std()))
    print("Max. loss: $%2.f" % (losses.min()))
    print("Min. loss: $%2.f" % (losses.max()))
    returns = tradesAnalyzer.getNegativeReturns()
    print("Avg. return: %2.f %%" % (returns.mean() * 100))
    print("Returns std. dev.: %2.f %%" % (returns.std() * 100))
    print("Max. return: %2.f %%" % (returns.max() * 100))
    print("Min. return: %2.f %%" % (returns.min() * 100))
```

The output should look like this:

```
Final portfolio value: $1295887.22
Cumulative returns: 29.59 %
Sharpe ratio: 0.70
Max. drawdown: 24.53 %
Longest drawdown duration: 277 days, 0:00:00

Total trades: 13
Avg. profit: $14437
Profits std. dev.: $127539
Max. profit: $420866
Min. profit: $-89320
Avg. return:  2 %
Returns std. dev.: 13 %
```

```
Max. return: 46 %
Min. return: -7 %

Profitable trades: 3
Avg. profit: $197053
Profits std. dev.: $158987
Max. profit: $420866
Min. profit: $66537
Avg. return: 21 %
Returns std. dev.: 18 %
Max. return: 46 %
Min. return:  6 %

Unprofitable trades: 10
Avg. loss: $-40348
Losses std. dev.: $23601
Max. loss: $-89320
Min. loss: $-4516
Avg. return: -3 %
Returns std. dev.:  2 %
Max. return: -0 %
Min. return: -7 %
```

plotter – Strategy plotter

optimizer – Parallel optimizers

注解:

- The server component will split strategy executions in chunks which are distributed among the different workers. **pyalgotrade.optimizer.server.Server.defaultBatchSize** controls the chunk size.
- The `pyalgotrade.strategy.BaseStrategy.getResult()` method is used to select the best strategy execution. You can override that method to rank executions using a different criteria.

marketsession – Market sessions

class `pyalgotrade.marketsession.`**`MarketSession`**

Bases: `object`

Base class for market sessions.

注解: This is a base class and should not be used directly.

classmethod **`getTimezone`**`()`

Returns the pytz timezone for the market session.

class `pyalgotrade.marketsession.`**`NASDAQ`**

Bases: *`pyalgotrade.marketsession.MarketSession`*

NASDAQ market session.

class `pyalgotrade.marketsession.`**`NYSE`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

New York Stock Exchange market session.

class `pyalgotrade.marketsession.`**`USEquities`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

US Equities market session.

class `pyalgotrade.marketsession.`**`MERVAL`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

Buenos Aires (Argentina) market session.

class `pyalgotrade.marketsession.`**`BOVESPA`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

BOVESPA (Brazil) market session.

class `pyalgotrade.marketsession.`**`FTSE`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

London Stock Exchange market session.

class `pyalgotrade.marketsession.`**`TSE`**
Bases: *`pyalgotrade.marketsession.MarketSession`*

Tokyo Stock Exchange market session.

CHAPTER 4

工具

Yahoo! Finance

Quandl

BarFeed resampling

CHAPTER 5

Event profiler

Inspired in QSTK (http://wiki.quantsoftware.org/index.php?title=QSTK_Tutorial_9), the **eventprofiler** module is a tool to analyze, statistically, how events affect future equity prices. The event profiler scans over historical data for a specified event and then calculates the impact of that event on the equity prices in the past and the future over a certain lookback period.

The goal of this tool is to help you quickly validate an idea, before moving forward with the backtesting process.

Example

The following example is inspired on the 'Buy-on-Gap Model' from Ernie Chan's book: 'Algorithmic Trading: Winning Strategies and Their Rationale':

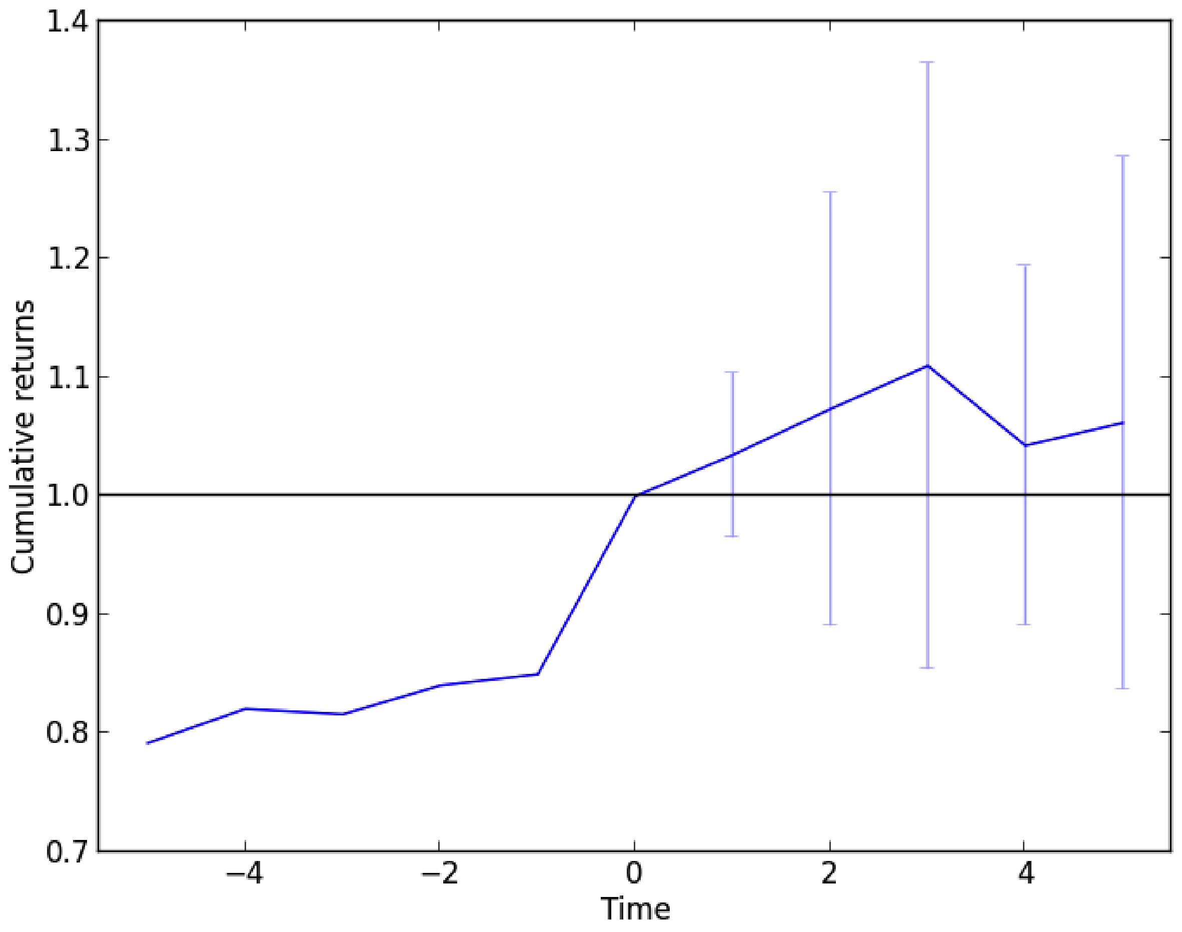
- The idea is to select a stock near the market open whose returns from their previous day's lows to today's open are lower that one standard deviation. The standard deviation is computed using the daily close-to-close returns of the last 90 days. These are the stocks that "gapped down".
- This is narrowed down by requiring the open price to be higher than the 20-day moving average of the closing price.

```
from pyalgotrade import eventprofiler
from pyalgotrade.technical import stats
from pyalgotrade.technical import roc
from pyalgotrade.technical import ma
from pyalgotrade.tools import yahoofinance

# Event inspired on an example from Ernie Chan's book:
# 'Algorithmic Trading: Winning Strategies and Their Rationale'


class BuyOnGap(eventprofiler.Predicate):
    def __init__(self, feed):
        stdDevPeriod = 90
        smaPeriod = 20
        self.__returns = {}
        self.__stdDev = {}
        self.__ma = {}
        for instrument in feed.getRegisteredInstruments():
            priceDS = feed[instrument].getAdjCloseDataSeries()
            # Returns over the adjusted close values.
            self.__returns[instrument] = roc.RateOfChange(priceDS, 1)
            # StdDev over those returns.
            self.__stdDev[instrument] = stats.StdDev(self.__returns[instrument],␣
↪stdDevPeriod)
            # MA over the adjusted close values.
            self.__ma[instrument] = ma.SMA(priceDS, smaPeriod)

    def __gappedDown(self, instrument, bards):
        ret = False
        if self.__stdDev[instrument][-1] is not None:
            prevBar = bards[-2]
            currBar = bards[-1]
            low2OpenRet = (currBar.getOpen(True) - prevBar.getLow(True)) /␣
↪float(prevBar.getLow(True))
            if low2OpenRet < (self.__returns[instrument][-1] - self.__
↪stdDev[instrument][-1]):
                ret = True
        return ret

    def __aboveSMA(self, instrument, bards):
        ret = False
        if self.__ma[instrument][-1] is not None and bards[-1].getOpen(True) > self.__
↪ma[instrument][-1]:
            ret = True
        return ret

    def eventOccurred(self, instrument, bards):
        ret = False
        if self.__gappedDown(instrument, bards) and self.__aboveSMA(instrument,␣
↪bards):
            ret = True
        return ret


def main(plot):
    instruments = ["AA", "AES", "AIG"]
    feed = yahoofinance.build_feed(instruments, 2008, 2009, ".")

    predicate = BuyOnGap(feed)
    eventProfiler = eventprofiler.Profiler(predicate, 5, 5)
    eventProfiler.run(feed, True)

    results = eventProfiler.getResults()
    print("%d events found" % (results.getEventCount()))
    if plot:
        eventprofiler.plot(results)

if __name__ == "__main__":
    main(True)
```

The code is doing 4 things:

1. Declaring a `Predicate` that implements the 'Buy-on-Gap Model' event identification.
2. Loading bars for some stocks.
3. Running the analysis.
4. Plotting the results.

This is what the output should look like:



```
2013-09-20 23:30:45,340 yahoofinance [INFO] Creating data directory
2013-09-20 23:30:45,341 yahoofinance [INFO] Downloading AA 2008 to data/AA-2008-
↪yahoofinance.csv
2013-09-20 23:30:46,092 yahoofinance [INFO] Downloading AES 2008 to data/AES-2008-
↪yahoofinance.csv
2013-09-20 23:30:46,683 yahoofinance [INFO] Downloading AIG 2008 to data/AIG-2008-
↪yahoofinance.csv
2013-09-20 23:30:47,260 yahoofinance [INFO] Downloading AA 2009 to data/AA-2009-
↪yahoofinance.csv
2013-09-20 23:30:48,019 yahoofinance [INFO] Downloading AES 2009 to data/AES-2009-
↪yahoofinance.csv
2013-09-20 23:30:48,761 yahoofinance [INFO] Downloading AIG 2009 to data/AIG-2009-
↪yahoofinance.csv
14 events found
```

Note that **Cummulative returns are normalized to the time of the event**.

CHAPTER 6

Xignite support

The xignite package adds support for paper trading strategies using Xignite's live feeds via the XigniteGlobalRealTime API (https://www.xignite.com/product/global-real-time-stock-quote-data/) in conjunction with PyAlgoTrade backtesting capabilities.

Contents:

xignite – Xignite reference

Feeds

Xignite Example

This goal of this example is to show how to put all the pieces together to paper trade a strategy using realtime feeds supplied by Xignite (https://www.xignite.com/).

This example assumes that you're already familiar with the basic concepts presented in the 简明教程 section.

The key things to highlight are:

1. We're using `pyalgotrade.strategy.BaseStrategy` instead of `pyalgotrade.strategy.BacktestingStrategy` as the base class. This is not a backtest.
2. `pyalgotrade.xignite.barfeed.LiveFeed` is used to pull 5 minute bars directly from Xignite. In order to use this service you need to sign up for the XigniteGlobalRealTime API (https://www.xignite.com/product/global-real-time-stock-quote-data/) to get an API token.
3. As described in https://www.xignite.com/product/global-real-time-stock-quote-data/api/GetBar/, indentifiers are fully qualified identifiers for the security as determined by the **IdentifierType** parameter. You **must** include the exchange suffix.
4. You can only paper trade while the market is open.

5. The 5 minute bars are requested 60 seconds after the 5 minute window closes because data may not be immediately available.

```
from pyalgotrade import strategy
from pyalgotrade.bar import Frequency
from pyalgotrade.xignite import barfeed
from pyalgotrade.broker import backtesting
from pyalgotrade.technical import ma


class Strategy(strategy.BaseStrategy):
    def __init__(self, feed, brk):
        strategy.BaseStrategy.__init__(self, feed, brk)
        self.__sma = {}
        for instrument in feed.getRegisteredInstruments():
            self.__sma[instrument] = ma.SMA(feed[instrument].getCloseDataSeries(), 5)

    def onBars(self, bars):
        for instrument in bars.getInstruments():
            bar = bars[instrument]
            self.info("%s: Open: %s High: %s Low: %s Close: %s Volume: %s SMA: %s" %␣
↪(instrument, bar.getOpen(), bar.getHigh(), bar.getLow(), bar.getClose(), bar.
↪getVolume(), self.__sma[instrument][-1]))


def main():
    # Replace apiToken with your own API token.
    apiToken = "<YOUR API TOKEN HERE>"
    # indentifiers are fully qualified identifiers for the security and must include␣
↪the exchange suffix.
    indentifiers = ["RIOl.CHIX", "HSBAl.CHIX"]
    # apiCallDelay is necessary because the bar may not be immediately available.
    apiCallDelay = 60

    feed = barfeed.LiveFeed(apiToken, indentifiers, Frequency.MINUTE*5, apiCallDelay)
    brk = backtesting.Broker(1000, feed)
    myStrategy = Strategy(feed, brk)
    myStrategy.run()

if __name__ == "__main__":
    main()
```

The output should look like this:

```
2014-03-28 09:31:01,389 strategy [INFO] HSBAl.CHIX: Open: 6.093 High: 6.102 Low: 6.
↪093 Close: 6.101 Volume: 45635.0 SMA: None
2014-03-28 09:31:01,390 strategy [INFO] RIOl.CHIX: Open: 33.08 High: 33.08 Low: 33.
↪065 Close: 33.07 Volume: 2303.0 SMA: None
2014-03-28 09:36:01,494 strategy [INFO] HSBAl.CHIX: Open: 6.102 High: 6.102 Low: 6.
↪099 Close: 6.099 Volume: 21043.0 SMA: None
2014-03-28 09:36:01,495 strategy [INFO] RIOl.CHIX: Open: 33.075 High: 33.09 Low: 33.
↪055 Close: 33.055 Volume: 2909.0 SMA: None
2014-03-28 09:41:01,885 strategy [INFO] HSBAl.CHIX: Open: 6.101 High: 6.101 Low: 6.
↪097 Close: 6.097 Volume: 29075.0 SMA: None
2014-03-28 09:41:01,886 strategy [INFO] RIOl.CHIX: Open: 33.04 High: 33.04 Low: 33.
↪005 Close: 33.005 Volume: 895.0 SMA: None
2014-03-28 09:46:00,943 strategy [INFO] HSBAl.CHIX: Open: 6.098 High: 6.098 Low: 6.
↪093 Close: 6.094 Volume: 17955.0 SMA: None
2014-03-28 09:46:00,943 strategy [INFO] RIOl.CHIX: Open: 33.005 High: 33.035 Low: 32.
↪995 Close: 32.995 Volume: 4052.0 SMA: None
```

```
2014-03-28 09:51:01,604 strategy [INFO] HSBAl.CHIX: Open: 6.093 High: 6.099 Low: 6.
↪092 Close: 6.097 Volume: 28046.0 SMA: 6.0976
2014-03-28 09:51:01,604 strategy [INFO] RIOl.CHIX: Open: 32.99 High: 33.025 Low: 32.
↪985 Close: 32.99 Volume: 2823.0 SMA: 33.023
2014-03-28 09:56:01,511 strategy [INFO] HSBAl.CHIX: Open: 6.099 High: 6.1 Low: 6.096␣
↪Close: 6.098 Volume: 19713.0 SMA: 6.097
2014-03-28 09:56:01,511 strategy [INFO] RIOl.CHIX: Open: 32.98 High: 33.01 Low: 32.96␣
↪Close: 33.01 Volume: 1545.0 SMA: 33.011
.
.
.
```

When apiCallDelay is not long enough, or when there is no data at all, you may receive the following error message:

```
xignite [ERROR] No ticks available for Symbol:RIOl.CHIX from 3/28/2014 1:10:00 PM to␣
↪3/28/2014 1:11:00 PM.
```

CHAPTER 7

Bitcoin

Contents:

Bitstamp support

The bitstamp package adds support for paper trading strategies using Bitstamp's live trade and orderbook feeds (https://www.bitstamp.net/websocket/) in conjunction with PyAlgoTrade backtesting capabilities. Future versions will support real trading.

Bitstamp support depends on **ws4py** (https://github.com/Lawouach/WebSocket-for-Python) and **tornado** (http://www.tornadoweb.org/en/stable/) so be sure to have those installed before moving forward.

Contents:

bitstamp – Bitstamp reference

WebSocket

This package has classes for the events emitted by Bitstamp's streaming service. Check https://www.bitstamp.net/websocket/ for more information.

Feeds

Brokers

Bitstamp Example

This goal of this simple SMA crossover example is to show how to put all the pieces together to paper trade a strategy using realtime feeds supplied by Bitstamp (https://www.bitstamp.net/).

This example assumes that you're already familiar with the basic concepts presented in the 简明教程 section.

The key things to highlight are:

1. We're using `pyalgotrade.strategy.BaseStrategy` instead of `pyalgotrade.strategy.BacktestingStrategy` as the base class. This is not a backtest.
2. Trade events get notified via the call to **onBars**. No need to manually subscribe.
3. Order book update events are handled by manually subscribing to `pyalgotrade.bitstamp.barfeed.LiveTradeFeed.getOrderBookUpdateEvent`. This is needed to be up to date with latest bid and ask prices.

```python
from pyalgotrade.bitstamp import barfeed
from pyalgotrade.bitstamp import broker
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import cross


class Strategy(strategy.BaseStrategy):
    def __init__(self, feed, brk):
        strategy.BaseStrategy.__init__(self, feed, brk)
        smaPeriod = 20
        self.__instrument = "BTC"
        self.__prices = feed[self.__instrument].getCloseDataSeries()
        self.__sma = ma.SMA(self.__prices, smaPeriod)
        self.__bid = None
        self.__ask = None
        self.__position = None
        self.__posSize = 0.05

        # Subscribe to order book update events to get bid/ask prices to trade.
        feed.getOrderBookUpdateEvent().subscribe(self.__onOrderBookUpdate)

    def __onOrderBookUpdate(self, orderBookUpdate):
        bid = orderBookUpdate.getBidPrices()[0]
        ask = orderBookUpdate.getAskPrices()[0]

        if bid != self.__bid or ask != self.__ask:
            self.__bid = bid
            self.__ask = ask
            self.info("Order book updated. Best bid: %s. Best ask: %s" % (self.__bid,␣
↪self.__ask))

    def onEnterOk(self, position):
        self.info("Position opened at %s" % (position.getEntryOrder().
↪getExecutionInfo().getPrice()))

    def onEnterCanceled(self, position):
        self.info("Position entry canceled")
        self.__position = None

    def onExitOk(self, position):
        self.__position = None
        self.info("Position closed at %s" % (position.getExitOrder().
↪getExecutionInfo().getPrice()))

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        self.__position.exitLimit(self.__bid)
    def onBars(self, bars):
        bar = bars[self.__instrument]
        self.info("Price: %s. Volume: %s." % (bar.getClose(), bar.getVolume()))

        # Wait until we get the current bid/ask prices.
        if self.__ask is None:
            return

        # If a position was not opened, check if we should enter a long position.
        if self.__position is None:
            if cross.cross_above(self.__prices, self.__sma) > 0:
                self.info("Entry signal. Buy at %s" % (self.__ask))
                self.__position = self.enterLongLimit(self.__instrument, self.__ask,␣
↪self.__posSize, True)
        # Check if we have to close the position.
        elif not self.__position.exitActive() and cross.cross_below(self.__prices,␣
↪self.__sma) > 0:
            self.info("Exit signal. Sell at %s" % (self.__bid))
            self.__position.exitLimit(self.__bid)


def main():
    barFeed = barfeed.LiveTradeFeed()
    brk = broker.PaperTradingBroker(1000, barFeed)
    strat = Strategy(barFeed, brk)

    strat.run()

if __name__ == "__main__":
    main()
```

The output should look like this:

```
2014-03-15 00:35:59,085 bitstamp [INFO] Initializing websocket client.
2014-03-15 00:35:59,452 bitstamp [INFO] Connection established.
2014-03-15 00:35:59,453 bitstamp [INFO] Initialization ok.
2014-03-15 00:36:30,726 strategy [INFO] Order book updated. Best bid: 629.6. Best␣
↪ask: 630.0
2014-03-15 00:39:04,829 strategy [INFO] Order book updated. Best bid: 628.89. Best␣
↪ask: 630.0
2014-03-15 00:44:18,845 strategy [INFO] Price: 630.0. Volume: 0.01.
2014-03-15 00:44:18,894 strategy [INFO] Order book updated. Best bid: 630.0. Best␣
↪ask: 631.49
2014-03-15 00:44:29,719 strategy [INFO] Price: 630.0. Volume: 0.02.
2014-03-15 00:44:59,861 strategy [INFO] Price: 631.49. Volume: 0.03360823.
2014-03-15 00:45:37,425 strategy [INFO] Order book updated. Best bid: 630.0. Best␣
↪ask: 631.6
2014-03-15 00:45:39,848 strategy [INFO] Price: 631.6. Volume: 3.35089782.
2014-03-15 00:45:39,918 strategy [INFO] Price: 632.24. Volume: 0.136.
2014-03-15 00:45:39,971 strategy [INFO] Price: 632.24. Volume: 0.138.
2014-03-15 00:45:40,057 strategy [INFO] Price: 632.25. Volume: 0.09076537.
2014-03-15 00:45:40,104 strategy [INFO] Price: 632.42. Volume: 0.74011681.
2014-03-15 00:45:40,205 strategy [INFO] Order book updated. Best bid: 630.0. Best␣
↪ask: 632.42
2014-03-15 00:48:30,005 strategy [INFO] Price: 630.0. Volume: 4.97.
2014-03-15 00:48:30,039 strategy [INFO] Price: 629.6. Volume: 0.09.
2014-03-15 00:48:30,121 strategy [INFO] Price: 629.54. Volume: 0.09.
```

```
.
.
.
2014-03-15 00:48:33,053 strategy [INFO] Price: 625.55. Volume: 1.296299.
2014-03-15 00:48:33,164 strategy [INFO] Price: 625.52. Volume: 0.0924981.
2014-03-15 00:48:33,588 strategy [INFO] Price: 625.45. Volume: 13.46260589.
2014-03-15 00:48:33,635 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 632.42
2014-03-15 00:48:33,727 strategy [INFO] Price: 625.45. Volume: 1.75.
2014-03-15 00:48:34,261 strategy [INFO] Price: 625.48. Volume: 0.1.
2014-03-15 00:48:34,908 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 631.39
2014-03-15 00:48:36,203 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 632.42
.
.
.
2014-03-15 00:49:01,945 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 631.39
2014-03-15 00:49:34,743 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 631.28
2014-03-15 00:49:57,651 strategy [INFO] Price: 629.26. Volume: 0.66893865.
2014-03-15 00:49:57,651 strategy [INFO] Entry signal. Buy at 631.28
2014-03-15 00:50:09,934 strategy [INFO] Order book updated. Best bid: 629.26. Best␣
↪ask: 631.39
2014-03-15 00:50:20,786 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 631.39
2014-03-15 00:50:25,658 strategy [INFO] Price: 631.39. Volume: 0.01.
2014-03-15 00:50:25,732 strategy [INFO] Price: 631.39. Volume: 0.01.
2014-03-15 00:50:25,791 strategy [INFO] Price: 631.93. Volume: 0.5.
2014-03-15 00:50:25,847 strategy [INFO] Price: 632.42. Volume: 0.25988319.
2014-03-15 00:50:25,900 strategy [INFO] Price: 632.42. Volume: 0.184.
2014-03-15 00:50:25,952 strategy [INFO] Price: 632.42. Volume: 0.184.
2014-03-15 00:50:26,000 strategy [INFO] Price: 632.42. Volume: 0.184.
2014-03-15 00:50:26,065 strategy [INFO] Price: 632.44. Volume: 0.184.
2014-03-15 00:50:26,139 strategy [INFO] Price: 632.97. Volume: 0.092.
2014-03-15 00:50:26,300 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 629.0
2014-03-15 00:50:26,398 strategy [INFO] Price: 633.1. Volume: 0.16211681.
2014-03-15 00:50:29,850 strategy [INFO] Position opened at 629.0
2014-03-15 00:50:29,850 strategy [INFO] Price: 629.0. Volume: 0.25152623.
2014-03-15 00:50:29,850 strategy [INFO] Exit signal. Sell at 627.02
2014-03-15 00:50:37,294 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 633.1
2014-03-15 00:50:43,526 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 633.08
2014-03-15 00:51:07,604 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 632.99
2014-03-15 00:51:46,194 strategy [INFO] Order book updated. Best bid: 627.02. Best␣
↪ask: 630.89
2014-03-15 00:52:08,223 strategy [INFO] Position closed at 627.02
2014-03-15 00:52:08,223 strategy [INFO] Price: 627.02. Volume: 0.0924979.
2014-03-15 00:52:08,290 strategy [INFO] Price: 627.02. Volume: 0.02.
2014-03-15 00:52:08,530 strategy [INFO] Order book updated. Best bid: 627.01. Best␣
↪ask: 627.02
2014-03-15 00:52:35,347 strategy [INFO] Price: 630.89. Volume: 0.07.
2014-03-15 00:52:35,347 strategy [INFO] Entry signal. Buy at 627.02
2014-03-15 00:52:35,348 strategy [INFO] Price: 630.95. Volume: 0.09.
```

```
2014-03-15 00:52:35,428 strategy [INFO] Price: 631.0. Volume: 0.05.
.
.
.
2014-03-15 00:54:14,077 strategy [INFO] Order book updated. Best bid: 628.81. Best␣
↪ask: 632.4
2014-03-15 00:54:21,084 strategy [INFO] Order book updated. Best bid: 631.0. Best␣
↪ask: 632.4
2014-03-15 00:54:34,484 strategy [INFO] Price: 632.4. Volume: 0.296299.
2014-03-15 00:54:34,484 strategy [INFO] Exit signal. Sell at 631.0
2014-03-15 00:54:34,484 strategy [INFO] Position entry canceled
2014-03-15 00:54:34,578 strategy [INFO] Price: 632.45. Volume: 3.5.
2014-03-15 00:54:34,642 strategy [INFO] Price: 632.45. Volume: 0.5715.
2014-03-15 00:54:34,708 strategy [INFO] Price: 632.46. Volume: 0.136.
2014-03-15 00:54:34,789 strategy [INFO] Price: 632.46. Volume: 1.2682.
2014-03-15 00:54:34,885 strategy [INFO] Price: 632.46. Volume: 3.5.
2014-03-15 00:54:34,949 strategy [INFO] Price: 632.46. Volume: 5.25.
2014-03-15 00:54:35,029 strategy [INFO] Price: 632.47. Volume: 9.88740834.
2014-03-15 00:54:35,165 strategy [INFO] Price: 632.89. Volume: 18.24259574.
2014-03-15 00:54:35,165 strategy [INFO] Entry signal. Buy at 632.4
2014-03-15 00:54:35,286 strategy [INFO] Price: 633.0. Volume: 0.092.
2014-03-15 00:54:35,357 strategy [INFO] Price: 633.1. Volume: 0.37612853.
.
.
.
2014-03-15 00:56:48,885 strategy [INFO] Order book updated. Best bid: 632.1. Best␣
↪ask: 634.35
2014-03-15 00:56:57,234 strategy [INFO] Position opened at 632.1
2014-03-15 00:56:57,235 strategy [INFO] Price: 632.1. Volume: 0.66267992.
2014-03-15 00:56:57,235 strategy [INFO] Exit signal. Sell at 632.1
2014-03-15 00:56:57,268 strategy [INFO] Price: 631.83. Volume: 59.33732008.
2014-03-15 00:56:57,356 strategy [INFO] Order book updated. Best bid: 631.83. Best␣
↪ask: 634.35
2014-03-15 00:57:03,528 strategy [INFO] Price: 631.83. Volume: 0.08267992.
2014-03-15 00:57:03,604 strategy [INFO] Order book updated. Best bid: 631.0. Best␣
↪ask: 631.83
2014-03-15 00:57:04,824 strategy [INFO] Order book updated. Best bid: 631.0. Best␣
↪ask: 634.34
2014-03-15 00:57:09,775 strategy [INFO] Order book updated. Best bid: 631.0. Best␣
↪ask: 634.33
2014-03-15 00:57:11,112 strategy [INFO] Order book updated. Best bid: 632.1. Best␣
↪ask: 634.33
2014-03-15 00:57:15,822 strategy [INFO] Position closed at 632.1
2014-03-15 00:57:15,822 strategy [INFO] Price: 632.1. Volume: 0.2.
2014-03-15 00:57:20,839 strategy [INFO] Price: 632.1. Volume: 0.46267992.
2014-03-15 00:57:22,065 strategy [INFO] Price: 631.2. Volume: 0.03732008.
2014-03-15 00:57:22,122 strategy [INFO] Price: 634.33. Volume: 0.135.
2014-03-15 00:57:22,122 strategy [INFO] Entry signal. Buy at 634.33
2014-03-15 00:57:22,184 strategy [INFO] Price: 634.34. Volume: 0.97972409.
.
.
.
```

In order to live trade this strategy you should use `pyalgotrade.bitstamp.broker.LiveBroker` instead of `pyalgotrade.bitstamp.broker.PaperTradingBroker`.

Note that if you try to live trade this strategy you will probably loose money. Before jumping to live trading, be sure to write your own strategy, backtest and paper trade it thoroughly before risking real money.

Bitcoin Charts support

The bitcoincharts package adds support for integrating with historical trade data supplied by http://www.bitcoincharts.com/ for backtesting Bitcoin strategies.

Historical trade data in CSV format is described in http://www.bitcoincharts.com/about/markets-api/, and files can be downloaded from http://api.bitcoincharts.com/v1/csv/.

Contents:

bitcoincharts – Bitcoin Charts reference

Feeds

Bitcoin Charts example

Although it is absolutely possible to backtest a strategy with tick data as supplied by http://www.bitcoincharts.com/about/markets-api/ using `pyalgotrade.bitcoincharts.barfeed.CSVTradeFeed`, you may want to to backtest using summarized bars at a different frequency to make backtesting faster.

As of 12-Aug-2014, http://api.bitcoincharts.com/v1/csv/bitstampUSD.csv.gz has 4588830 events so we'll transform a portion of it into 30 minute bars for backtesting purposes with the following script:

```
from pyalgotrade.bitcoincharts import barfeed
from pyalgotrade.tools import resample
from pyalgotrade import bar

import datetime


def main():
    barFeed = barfeed.CSVTradeFeed()
    barFeed.addBarsFromCSV("bitstampUSD.csv", fromDateTime=datetime.datetime(2014, 1,␣
↪1))
    resample.resample_to_csv(barFeed, bar.Frequency.MINUTE*30, "30min-bitstampUSD.csv
↪")


if __name__ == "__main__":
    main()
```

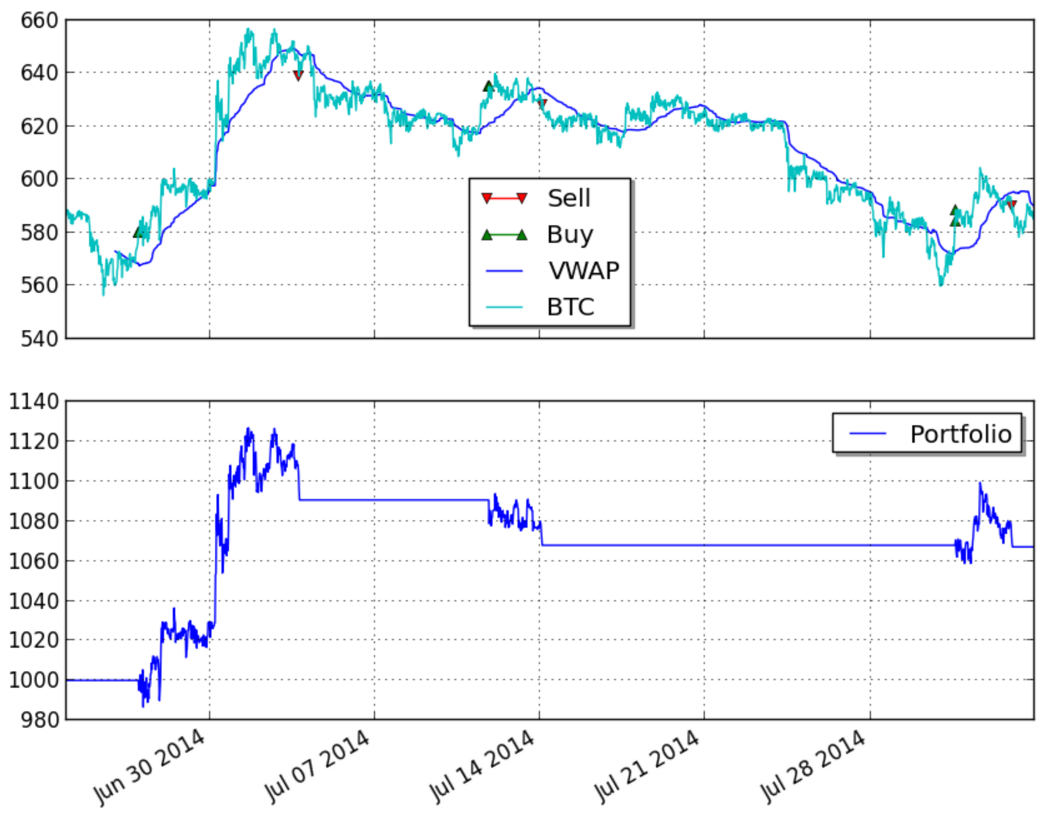
It will take some time to execute, so be patient. The resampled file should look like this:

```
Date Time,Open,High,Low,Close,Volume,Adj Close
2014-01-01 00:00:00,732.0,738.25,729.01,734.81,266.17955488,
2014-01-01 00:30:00,734.81,739.9,734.47,739.02,308.96802502,
2014-01-01 01:00:00,739.02,739.97,737.65,738.11,65.66924473,
2014-01-01 01:30:00,738.0,742.0,737.65,741.89,710.27165024,
2014-01-01 02:00:00,741.89,757.99,741.89,752.23,1085.13335011,
2014-01-01 02:30:00,752.23,755.0,747.0,747.2,272.03949342,
2014-01-01 04:00:00,744.98,748.02,744.98,747.19,104.65989075,
.
.
```

We can now take advantage of `pyalgotrade.barfeed.csvfeed.GenericBarFeed` to load the resampled file and backtest a Bitcoin strategy. We'll be using a VWAP momentum strategy for illustration purposes:

```
from pyalgotrade import bar
from pyalgotrade import strategy
from pyalgotrade import plotter
from pyalgotrade.technical import vwap
from pyalgotrade.barfeed import csvfeed
from pyalgotrade.bitstamp import broker
from pyalgotrade import broker as basebroker


class VWAPMomentum(strategy.BacktestingStrategy):
    MIN_TRADE = 5

    def __init__(self, feed, brk, instrument, vwapWindowSize, buyThreshold,␣
↪sellThreshold):
        strategy.BacktestingStrategy.__init__(self, feed, brk)
        self.__instrument = instrument
        self.__vwap = vwap.VWAP(feed[instrument], vwapWindowSize)
        self.__buyThreshold = buyThreshold
        self.__sellThreshold = sellThreshold

    def _getActiveOrders(self):
        orders = self.getBroker().getActiveOrders()
        buy = [o for o in orders if o.isBuy()]
        sell = [o for o in orders if o.isSell()]
        return buy, sell

    def _cancelOrders(self, orders):
        brk = self.getBroker()
        for o in orders:
            self.info("Canceling order %s" % (o.getId()))
            brk.cancelOrder(o)

    def _buySignal(self, price):
        buyOrders, sellOrders = self._getActiveOrders()
        self._cancelOrders(sellOrders)

        brk = self.getBroker()
        cashAvail = brk.getCash() * 0.98
        size = round(cashAvail / price, 3)
        if len(buyOrders) == 0 and price*size > VWAPMomentum.MIN_TRADE:
            self.info("Buy %s at %s" % (size, price))
            try:
                self.limitOrder(self.__instrument, price, size)
            except Exception as e:
                self.error("Failed to buy: %s" % (e))

    def _sellSignal(self, price):
        buyOrders, sellOrders = self._getActiveOrders()
        self._cancelOrders(buyOrders)

        brk = self.getBroker()
        shares = brk.getShares(self.__instrument)
        if len(sellOrders) == 0 and shares > 0:
            self.info("Sell %s at %s" % (shares, price))
            self.limitOrder(self.__instrument, price, shares*-1)

    def getVWAP(self):
        return self.__vwap
    def onBars(self, bars):
        vwap = self.__vwap[-1]
        if vwap is None:
            return

        price = bars[self.__instrument].getClose()
        if price > vwap * (1 + self.__buyThreshold):
            self._buySignal(price)
        elif price < vwap * (1 - self.__sellThreshold):
            self._sellSignal(price)

    def onOrderUpdated(self, order):
        if order.isBuy():
            orderType = "Buy"
        else:
            orderType = "Sell"
        self.info("%s order %d updated - Status: %s - %s" % (
            orderType,
            order.getId(),
            basebroker.Order.State.toString(order.getState()),
            order.getExecutionInfo(),
        ))


def main(plot):
    instrument = "BTC"
    initialCash = 1000
    vwapWindowSize = 100
    buyThreshold = 0.02
    sellThreshold = 0.01

    barFeed = csvfeed.GenericBarFeed(bar.Frequency.MINUTE*30)
    barFeed.addBarsFromCSV(instrument, "30min-bitstampUSD.csv")
    brk = broker.BacktestingBroker(initialCash, barFeed)
    strat = VWAPMomentum(barFeed, brk, instrument, vwapWindowSize, buyThreshold,␣
↪sellThreshold)

    if plot:
        plt = plotter.StrategyPlotter(strat)
        plt.getInstrumentSubplot(instrument).addDataSeries("VWAP", strat.getVWAP())

    strat.run()

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

This is what the plot looks like:


660
640
620
600
580
560
540
Sell
Buy
VWAP
BTC
1140
1120
1100
1080
1060
1040
1020
1000
980
Portfolio
Jun 30 2014
Jul 07 2014
Jul 14 2014
Jul 21 2014
Jul 28 2014

CHAPTER 8

Twitter support

The twitter package adds support for receiving Twitter events in your strategy and incorporate those in your trading decisions. Note that since this is a realtime feed, it only makes sense in paper trading or real trading scenarios, but not in backtesting.

Twitter support depends on **tweepy** (https://github.com/tweepy/tweepy) so be sure to have it installed before moving forward.

Contents:

twitter – Twitter feed reference

Feed

Twitter Example

This goal of this simple example is to show you how to put all the pieces together to incorporate Twitter events in a strategy. We will be using Bitstamp's live feed since backtesting with Twitter is not supported so please take a look at the *Bitstamp Example* section before moving forward.

In order to connect to Twitter's API you'll need:

- Consumer key
- Consumer secret
- Access token
- Access token secret

Go to http://dev.twitter.com and create an app. The consumer key and secret will be generated for you after that. Then you'll need to create an access token under the "Your access token" section.

The key things to highlight are:

1. We're using `pyalgotrade.strategy.BaseStrategy` instead of `pyalgotrade.strategy.BacktestingStrategy` as the base class. This is not a backtest.
2. The `pyalgotrade.twitter.feed.TwitterFeed` instance has to be included in the strategy event dispatch loop before running the strategy.

```
from pyalgotrade import strategy
from pyalgotrade.bitstamp import barfeed
from pyalgotrade.bitstamp import broker
from pyalgotrade.twitter import feed as twitterfeed


class Strategy(strategy.BaseStrategy):
    def __init__(self, feed, brk, twitterFeed):
        strategy.BaseStrategy.__init__(self, feed, brk)
        self.__instrument = "BTC"

        # Subscribe to Twitter events.
        twitterFeed.subscribe(self.__onTweet)

    def __onTweet(self, data):
        # Refer to https://dev.twitter.com/docs/streaming-apis/messages#Public_stream_
↪messages for
        # the information available in data.
        try:
            self.info("Twitter: %s" % (data["text"]))
        except KeyError:
            pass

    def onBars(self, bars):
        bar = bars[self.__instrument]
        self.info("Price: %s. Volume: %s." % (bar.getClose(), bar.getVolume()))


def main():
    # Go to http://dev.twitter.com and create an app.
    # The consumer key and secret will be generated for you after that.
    consumer_key = "<YOUR-CONSUMER-KEY-HERE>"
    consumer_secret = "<YOUR-CONSUMER-SECRET-HERE>"

    # After the step above, you will be redirected to your app's page.
    # Create an access token under the the "Your access token" section
    access_token = "<YOUR-ACCESS-TOKEN-HERE>"
    access_token_secret = "<YOUR-ACCESS-TOKEN-SECRET-HERE>"

    # Create a twitter feed to track BitCoin related events.
    track = ["bitcoin", "btc", "mtgox", "bitstamp", "xapo"]
    follow = []
    languages = ["en"]
    twitterFeed = twitterfeed.TwitterFeed(consumer_key, consumer_secret, access_token,
↪ access_token_secret, track, follow, languages)

    barFeed = barfeed.LiveTradeFeed()
    brk = broker.PaperTradingBroker(1000, barFeed)
    strat = Strategy(barFeed, brk, twitterFeed)

    # It is VERY important to add twitterFeed to the event dispatch loop before␣
↪running the strategy.
    strat.getDispatcher().addSubject(twitterFeed)
    strat.run()

if __name__ == "__main__":
    main()
```

The output should look like this:

```
2014-03-15 02:42:07,696 bitstamp [INFO] Initializing client.
2014-03-15 02:42:08,174 bitstamp [INFO] Connection established.
2014-03-15 02:42:08,175 bitstamp [INFO] Initialization ok.
2014-03-15 02:42:08,175 twitter [INFO] Initializing client.
2014-03-15 02:42:09,238 twitter [INFO] Connected.
2014-03-15 02:42:15,107 strategy [INFO] Twitter: Warren Buffett Urges Investors to␣
↪'Stay Away' from Bitcoin http://t.co/WWRCr3rfob
2014-03-15 02:42:22,926 strategy [INFO] Twitter: FundingUnion Inc. Bitcoin MLM Social␣
↪Build  - The Currently in BETA version - Join Today : http://t.co/bxcfoBV0I4
↪#Bitcoin #MLM #SocialMedia
2014-03-15 02:42:23,577 strategy [INFO] Twitter: #News Alleged Bitcoin creator denies␣
↪he's the one http://t.co/gmil085smQ #DailyNews
2014-03-15 02:42:29,378 strategy [INFO] Twitter: RT @Tom_Cruis3: #News Alleged␣
↪Bitcoin creator denies he's the one http://t.co/gmil085smQ #DailyNews
2014-03-15 02:42:38,012 strategy [INFO] Twitter: Need for speed dulu at BTC XXI
2014-03-15 02:42:40,162 strategy [INFO] Price: 632.97. Volume: 0.51896614.
2014-03-15 02:42:45,867 strategy [INFO] Twitter: Analysis of the leaked MTGOX␣
↪database http://t.co/6ty8EupGh9 via @Reddit
2014-03-15 02:42:46,761 strategy [INFO] Twitter: [ANN] After MtGox implosion, this is␣
↪the first exchange to implement transparent cold storage http://t.co/EyafH5jXsM
↪#reddit #bitcoin
2014-03-15 02:42:46,825 strategy [INFO] Twitter: @abatalion @naval @cdixon you guys␣
↪remember CentMail? http://t.co/iihgCS4OwJ would be interesting with a BTC twist.
2014-03-15 02:42:47,285 strategy [INFO] Twitter: Help me convince my Dad to mine␣
↪Bitcoins! http://t.co/fxboBooINu #reddit #bitcoin
2014-03-15 02:42:47,492 strategy [INFO] Twitter: RT @VentureBeat: Warren Buffett:␣
↪Bitcoin is a 'mirage' http://t.co/B1FcvZKJQS by @xBarryLevine
2014-03-15 02:42:47,625 strategy [INFO] Twitter: I believe in bitcoin as not only a␣
↪easy exchange of value but an 'anti bank movement' with occupy wall street,␣
↪environmentalists, cons...
2014-03-15 02:42:47,979 strategy [INFO] Twitter: Bitcoin protocol/algorithm is being␣
↪managed by a bunch of people. Isn't that a vulnerability? http://t.co/HJaCkgC6zT
↪#reddit #bitcoin
2014-03-15 02:42:48,388 strategy [INFO] Twitter: Does anyone still think Dorian␣
↪Nakomoto is Satoshi Nakamoto? http://t.co/v2N8SEZzxC #reddit #bitcoin
2014-03-15 02:42:53,313 strategy [INFO] Twitter: Last sale price at #Bitstamp was
↪$632.99 per #Bitcoin - Free SMS price alerts from 10 exchanges at http://t.co/
↪jsM3JqlvZd
2014-03-15 02:42:54,058 strategy [INFO] Twitter: Last sale price at #BTCe was $623.00␣
↪per #Bitcoin - Free SMS price alerts from 10 exchanges at http://t.co/jsM3JqlvZd
2014-03-15 02:42:54,059 strategy [INFO] Twitter: Last sale price at #Coinbase was
↪$632.99 per #Bitcoin - Free SMS price alerts from 10 exchanges at http://t.co/
↪jsM3JqlvZd
2014-03-15 02:42:57,148 strategy [INFO] Twitter: GET PAID WITH USD OR BITCOIN TO␣
↪PROMOTE LINKS!!!... http://t.co/HKANbuZQHM
2014-03-15 02:43:00,845 strategy [INFO] Twitter: We're back! Bitcoin expert Pete␣
↪Watson explaining just how the incredible #cryptocurrency works #bitcoin #HongKong␣
↪http://t.co/CJpTA6RujJ
2014-03-15 02:43:28,001 strategy [INFO] Twitter: Bitcoin's COO Explains What #Bitcoin␣
↪Is http://t.co/cQ1rVxYSUJ via @YouTube
```

```
2014-03-15 02:43:31,888 strategy [INFO] Twitter: #BINARY-OPTIONS! and BITCOIN! Trade␣
↪Area! With fantastic profits of up to 1,000% per trade and even more. @ http://t.co/
↪R9c9J4Q5VA
2014-03-15 02:43:42,148 strategy [INFO] Twitter: GET PAID WITH USD OR BITCOIN TO␣
↪PROMOTE LINKS!!!... http://t.co/bD6cacFKTs
```

CHAPTER 9

TA-Lib integration

The **pyalgotrade.talibext.indicator** module provides integration with Python wrapper for TA-Lib (http://mrjbq7.github.com/ta-lib/) to enable calling TA-Lib functions directly with `pyalgotrade.dataseries.DataSeries` or `pyalgotrade.dataseries.bards.BarDataSeries` instances instead of numpy arrays.

If you're familiar with the **talib** module, then using the **pyalgotrade.talibext.indicator** module should be straightforward. When using **talib** standalone you do something like this:

```
import numpy
import talib

data = numpy.random.random(100)
upper, middle, lower = talib.BBANDS(data, matype=talib.MA_T3)
```

To use the **pyalgotrade.talibext.indicator** module in your strategies you should do something like this:

```
def onBars(self, bars):
    closeDs = self.getFeed().getDataSeries("orcl").getCloseDataSeries()
    upper, middle, lower = pyalgotrade.talibext.indicator.BBANDS(closeDs, 100,␣
↪matype=talib.MA_T3)
    if upper != None:
        print "%s" % upper[-1]
```

Every function in the **pyalgotrade.talibext.indicator** module receives one or more dataseries (most receive just one) and the number of values to use from the dataseries. In the example above, we're calculating Bollinger Bands over the last 100 closing prices.

If the parameter name is **ds**, then you should pass a regular `pyalgotrade.dataseries.DataSeries` instance, like the one shown in the example above.

If the parameter name is **barDs**, then you should pass a `pyalgotrade.dataseries.bards.BarDataSeries` instance, like in the next example:

```
def onBars(self, bars):
    barDs = self.getFeed().getDataSeries("orcl")
    sar = indicator.SAR(barDs, 100)
    if sar != None:
        print "%s" % sar[-1]
```

The following TA-Lib functions are available through the **pyalgotrade.talibext.indicator** module:

CHAPTER 10

Computational Investing Part I

As I was taking the Computational Investing Part I course in 2012 I had to work on a set of assignments and for some of them I used PyAlgoTrade.

Homework 1

For this assignment I had to pick 4 stocks, invest a total of $100000 during 2011, and calculate:

- Final portfolio value
- Anual return
- Average daily return
- Std. dev. of daily returns
- Sharpe ratio

Download the data with the following commands:

```
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('aeti', 2011, 'aeti-2011-yahoofinance.csv')"
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('egan', 2011, 'egan-2011-yahoofinance.csv')"
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('glng', 2011, 'glng-2011-yahoofinance.csv')"
python -c "from pyalgotrade.tools import yahoofinance; yahoofinance.download_daily_
↪bars('simo', 2011, 'simo-2011-yahoofinance.csv')"
```

Although the deliverable was an Excel spreadsheet, I validated the results using this piece of code:

```
from pyalgotrade import strategy
from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.stratanalyzer import returns
from pyalgotrade.stratanalyzer import sharpe
from pyalgotrade.utils import stats
```

```
class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed):
        strategy.BacktestingStrategy.__init__(self, feed, 1000000)

        # We wan't to use adjusted close prices instead of close.
        self.setUseAdjustedValues(True)

        # Place the orders to get them processed on the first bar.
        orders = {
            "aeti": 297810,
            "egan": 81266,
            "glng": 11095,
            "simo": 17293,
        }
        for instrument, quantity in list(orders.items()):
            self.marketOrder(instrument, quantity, onClose=True, allOrNone=True)

    def onBars(self, bars):
        pass

# Load the yahoo feed from CSV files.
feed = yahoofeed.Feed()
feed.addBarsFromCSV("aeti", "aeti-2011-yahoofinance.csv")
feed.addBarsFromCSV("egan", "egan-2011-yahoofinance.csv")
feed.addBarsFromCSV("glng", "glng-2011-yahoofinance.csv")
feed.addBarsFromCSV("simo", "simo-2011-yahoofinance.csv")

# Evaluate the strategy with the feed's bars.
myStrategy = MyStrategy(feed)

# Attach returns and sharpe ratio analyzers.
retAnalyzer = returns.Returns()
myStrategy.attachAnalyzer(retAnalyzer)
sharpeRatioAnalyzer = sharpe.SharpeRatio()
myStrategy.attachAnalyzer(sharpeRatioAnalyzer)

# Run the strategy
myStrategy.run()

# Print the results.
print("Final portfolio value: $%.2f" % myStrategy.getResult())
print("Anual return: %.2f %%" % (retAnalyzer.getCumulativeReturns()[-1] * 100))
print("Average daily return: %.2f %%" % (stats.mean(retAnalyzer.getReturns()) * 100))
print("Std. dev. daily return: %.4f" % (stats.stddev(retAnalyzer.getReturns())))
print("Sharpe ratio: %.2f" % (sharpeRatioAnalyzer.getSharpeRatio(0)))
```

The results were:

```
Final portfolio value: $1604979.11
Anual return: 60.50 %
Average daily return: 0.20 %
Std. dev. daily return: 0.0136
Sharpe ratio: 2.30
```

Homework 3 and 4

For these assignments I had to build a market simulation tool that loads orders from a file, executes those, and prints the results for each day.

The orders file for homework 3 look like this:

```
2011,1,10,AAPL,Buy,1500,
2011,1,13,AAPL,Sell,1500,
2011,1,13,IBM,Buy,4000,
2011,1,26,GOOG,Buy,1000,
2011,2,2,XOM,Sell,4000,
2011,2,10,XOM,Buy,4000,
2011,3,3,GOOG,Sell,1000,
2011,3,3,IBM,Sell,2200,
2011,6,3,IBM,Sell,3300,
2011,5,3,IBM,Buy,1500,
2011,6,10,AAPL,Buy,1200,
2011,8,1,GOOG,Buy,55,
2011,8,1,GOOG,Sell,55,
2011,12,20,AAPL,Sell,1200,
```

This is the market simulation tool that I built:

```
import csv
import datetime
import os

from pyalgotrade.barfeed import yahoofeed
from pyalgotrade.barfeed import csvfeed
from pyalgotrade import strategy
from pyalgotrade.utils import stats
from pyalgotrade.stratanalyzer import returns
from pyalgotrade.stratanalyzer import sharpe


class OrdersFile:
    def __init__(self, ordersFile):
        self.__orders = {}
        self.__firstDate = None
        self.__lastDate = None
        self.__instruments = []

        # Load orders from the file.
        reader = csv.DictReader(open(ordersFile, "r"), fieldnames=["year", "month",
↪"day", "symbol", "action", "qty"])
        for row in reader:
            dateTime = datetime.datetime(int(row["year"]), int(row["month"]), int(row[
↪"day"]))
            self.__orders.setdefault(dateTime, [])
            order = (row["symbol"], row["action"], int(row["qty"]))
            self.__orders[dateTime].append(order)

            # As we process the file, store instruments, first date, and last date.
            if row["symbol"] not in self.__instruments:
                self.__instruments.append(row["symbol"])

            if self.__firstDate is None:
                self.__firstDate = dateTime
            else:
                self.__firstDate = min(self.__firstDate, dateTime)

            if self.__lastDate is None:
                self.__lastDate = dateTime
            else:
                self.__lastDate = max(self.__lastDate, dateTime)

    def getFirstDate(self):
        return self.__firstDate

    def getLastDate(self):
        return self.__lastDate

    def getInstruments(self):
        return self.__instruments

    def getOrders(self, dateTime):
        return self.__orders.get(dateTime, [])


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, cash, ordersFile, useAdjustedClose):
        # Suscribe to the feed bars event before the broker just to place the orders␣
↪properly.
        feed.getNewValuesEvent().subscribe(self.__onBarsBeforeBroker)
        strategy.BacktestingStrategy.__init__(self, feed, cash)
        self.__ordersFile = ordersFile
        self.setUseAdjustedValues(useAdjustedClose)
        # We will allow buying more shares than cash allows.
        self.getBroker().setAllowNegativeCash(True)

    def __onBarsBeforeBroker(self, dateTime, bars):
        for instrument, action, quantity in self.__ordersFile.getOrders(dateTime):
            if action.lower() == "buy":
                self.marketOrder(instrument, quantity, onClose=True)
            else:
                self.marketOrder(instrument, quantity*-1, onClose=True)

    def onOrderUpdated(self, order):
        if order.isCanceled():
            raise Exception("Order canceled. Ran out of cash ?")

    def onBars(self, bars):
        portfolioValue = self.getBroker().getEquity()
        self.info("Portfolio value: $%.2f" % (portfolioValue))


def main():
    # Load the orders file.
    ordersFile = OrdersFile("orders.csv")
    print("First date", ordersFile.getFirstDate())
    print("Last date", ordersFile.getLastDate())
    print("Symbols", ordersFile.getInstruments())

    # Load the data from QSTK storage. QS environment variable has to be defined.
    if os.getenv("QS") is None:
        raise Exception("QS environment variable not defined")
    feed = yahoofeed.Feed()
    feed.setBarFilter(csvfeed.DateRangeFilter(ordersFile.getFirstDate(), ordersFile.
↪getLastDate()))
    feed.setDailyBarTime(datetime.time(0, 0, 0))  # This is to match the dates loaded␣
↪with the ones in the orders file.
    for symbol in ordersFile.getInstruments():
        feed.addBarsFromCSV(symbol, os.path.join(os.getenv("QS"), "QSData", "Yahoo",␣
↪symbol + ".csv"))

    # Run the strategy.
    cash = 1000000
    useAdjustedClose = True
    myStrategy = MyStrategy(feed, cash, ordersFile, useAdjustedClose)

    # Attach returns and sharpe ratio analyzers.
    retAnalyzer = returns.Returns()
    myStrategy.attachAnalyzer(retAnalyzer)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    myStrategy.attachAnalyzer(sharpeRatioAnalyzer)

    myStrategy.run()

    # Print the results.
    print("Final portfolio value: $%.2f" % myStrategy.getResult())
    print("Anual return: %.2f %%" % (retAnalyzer.getCumulativeReturns()[-1] * 100))
    print("Average daily return: %.2f %%" % (stats.mean(retAnalyzer.getReturns()) *␣
↪100))
    print("Std. dev. daily return: %.4f" % (stats.stddev(retAnalyzer.getReturns())))
    print("Sharpe ratio: %.2f" % (sharpeRatioAnalyzer.getSharpeRatio(0)))

main()
```

The output for homework 3 looks like this:

```
First date 2011-01-10 00:00:00
Last date 2011-12-20 00:00:00
Symbols ['AAPL', 'IBM', 'GOOG', 'XOM']
2011-01-10 00:00:00: Portfolio value: $1000000.00
2011-01-11 00:00:00: Portfolio value: $998785.00
2011-01-12 00:00:00: Portfolio value: $1002940.00
2011-01-13 00:00:00: Portfolio value: $1004815.00
.
.
.
2011-12-15 00:00:00: Portfolio value: $1113532.00
2011-12-16 00:00:00: Portfolio value: $1116016.00
2011-12-19 00:00:00: Portfolio value: $1117444.00
2011-12-20 00:00:00: Portfolio value: $1133860.00
Final portfolio value: $1133860.00
Anual return: 13.39 %
Average daily return: 0.05 %
Std. dev. daily return: 0.0072
Sharpe ratio: 1.21
```

CHAPTER 11

策略例子

动量交易

VWAP动量交易

本例基于:

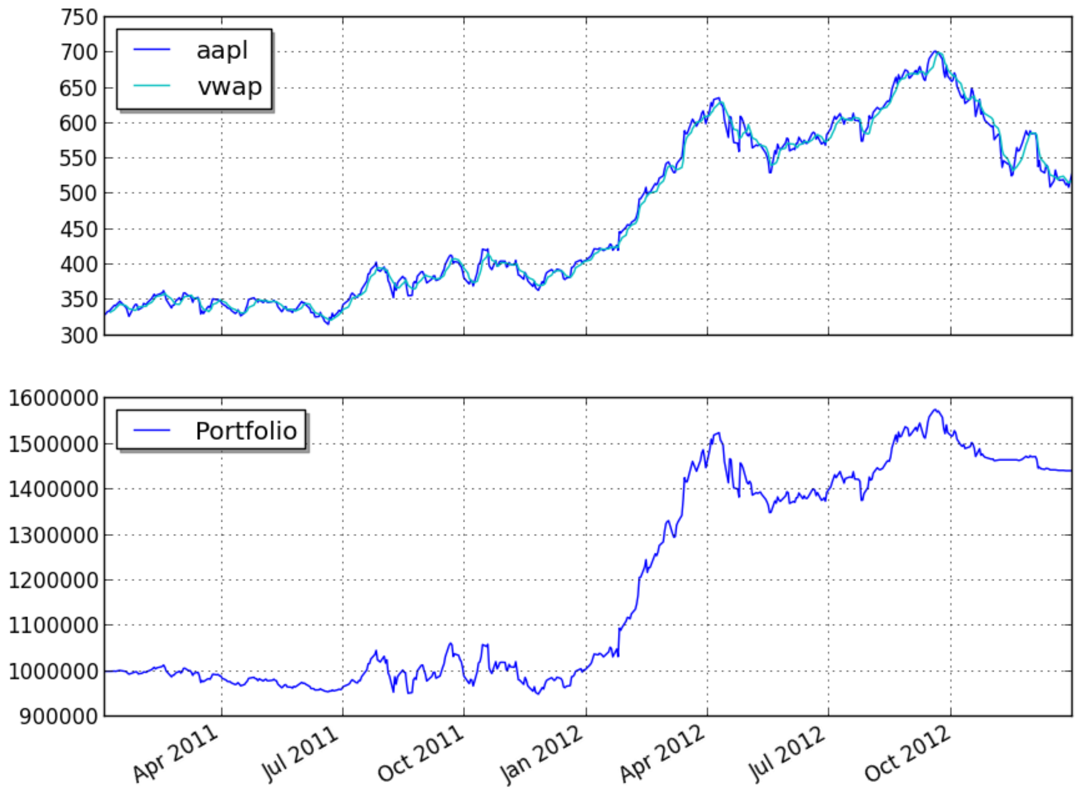
- https://www.quantopian.com/posts/momentum-trade

```
from pyalgotrade import strategy
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.technical import vwap
from pyalgotrade.stratanalyzer import sharpe


class VWAPMomentum(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, vwapWindowSize, threshold):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        self.__vwap = vwap.VWAP(feed[instrument], vwapWindowSize)
        self.__threshold = threshold

    def getVWAP(self):
        return self.__vwap

    def onBars(self, bars):
        vwap = self.__vwap[-1]
        if vwap is None:
            return

        shares = self.getBroker().getShares(self.__instrument)
        price = bars[self.__instrument].getClose()
        notional = shares * price
        if price > vwap * (1 + self.__threshold) and notional < 1000000:
            self.marketOrder(self.__instrument, 100)
        elif price < vwap * (1 - self.__threshold) and notional > 0:
            self.marketOrder(self.__instrument, -100)


def main(plot):
    instrument = "aapl"
    vwapWindowSize = 5
    threshold = 0.01

    # Download the bars.
    feed = yahoofinance.build_feed([instrument], 2011, 2012, ".")

    strat = VWAPMomentum(feed, instrument, vwapWindowSize, threshold)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, True, False, True)
        plt.getInstrumentSubplot(instrument).addDataSeries("vwap", strat.getVWAP())

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))

    if plot:
        plt.plot()

if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2013-09-21 00:01:23,813 yahoofinance [INFO] Creating data directory
2013-09-21 00:01:23,814 yahoofinance [INFO] Downloading aapl 2011 to data/aapl-2011-
↪yahoofinance.csv
2013-09-21 00:01:25,275 yahoofinance [INFO] Downloading aapl 2012 to data/aapl-2012-
↪yahoofinance.csv
Sharpe ratio: 0.89
```

最终结果绘制如下图所示:



你可以通过调整VWAP和参数阈值得到更高的回报。

SMA 交叉

保存代码到sma_crossover.py:

```
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import cross


class SMACrossOver(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, smaPeriod):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        self.__position = None
        # We'll use adjusted close values instead of regular close values.
        self.setUseAdjustedValues(True)
        self.__prices = feed[instrument].getPriceDataSeries()
        self.__sma = ma.SMA(self.__prices, smaPeriod)

    def getSMA(self):
        return self.__sma

    def onEnterCanceled(self, position):
        self.__position = None

    def onExitOk(self, position):
        self.__position = None

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        self.__position.exitMarket()

    def onBars(self, bars):
        # If a position was not opened, check if we should enter a long position.
        if self.__position is None:
            if cross.cross_above(self.__prices, self.__sma) > 0:
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                # Enter a buy market order. The order is good till canceled.
                self.__position = self.enterLong(self.__instrument, shares, True)
        # Check if we have to exit the position.
        elif not self.__position.exitActive() and cross.cross_below(self.__prices,␣
↪self.__sma) > 0:
            self.__position.exitMarket()
```

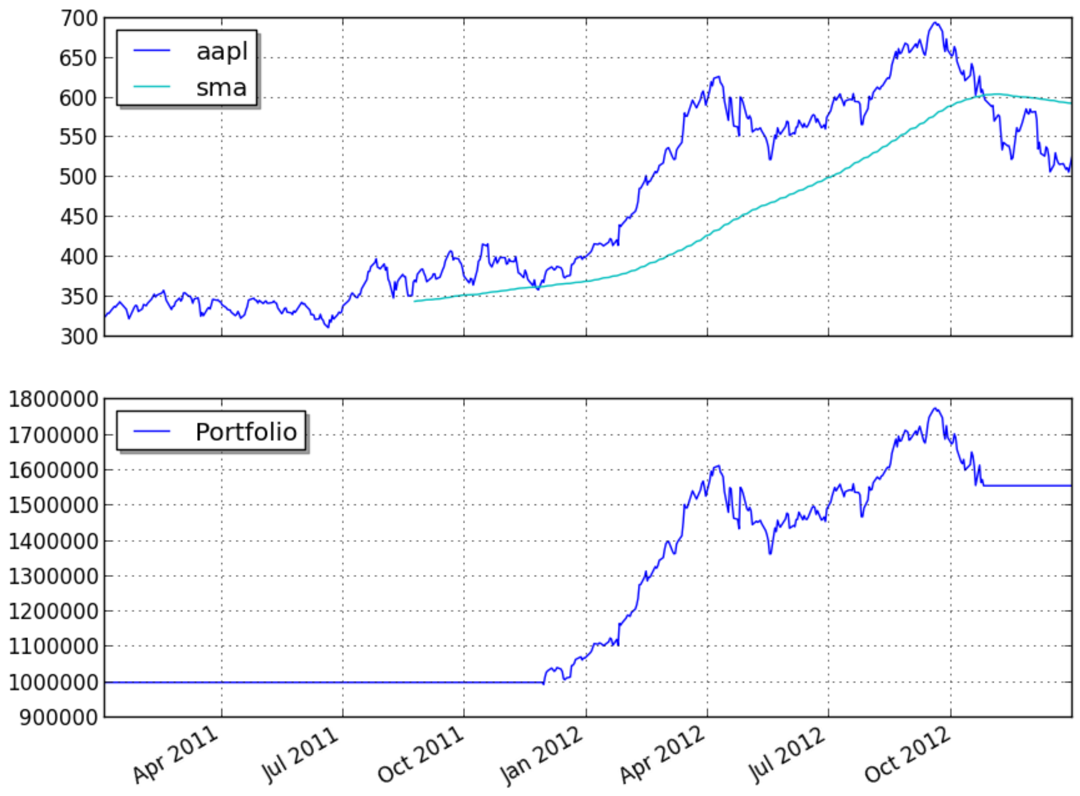
通过以下代码运行策略:

```
import sma_crossover
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.stratanalyzer import sharpe


def main(plot):
    instrument = "aapl"
    smaPeriod = 163

    # Download the bars.
    feed = yahoofinance.build_feed([instrument], 2011, 2012, ".")

    strat = sma_crossover.SMACrossOver(feed, instrument, smaPeriod)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, True, False, True)
        plt.getInstrumentSubplot(instrument).addDataSeries("sma", strat.getSMA())

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2013-09-21 00:01:23,813 yahoofinance [INFO] Creating data directory
2013-09-21 00:01:23,814 yahoofinance [INFO] Downloading aapl 2011 to data/aapl-2011-
↪yahoofinance.csv
2013-09-21 00:01:25,275 yahoofinance [INFO] Downloading aapl 2012 to data/aapl-2012-
↪yahoofinance.csv
Sharpe ratio: 1.12
```

最终结果绘制如下图所示:



你可以通过调整sma的周期得到更好的回报。

均线交叉择时

This example is inspired on the Market Timing / GTAA model described in:

- http://mebfaber.com/timing-model/
- http://papers.ssrn.com/sol3/papers.cfm?abstract_id=962461

The stragegy supports analyzing more than one instrument per asset class, and selects the one that has highest returns in the last month.

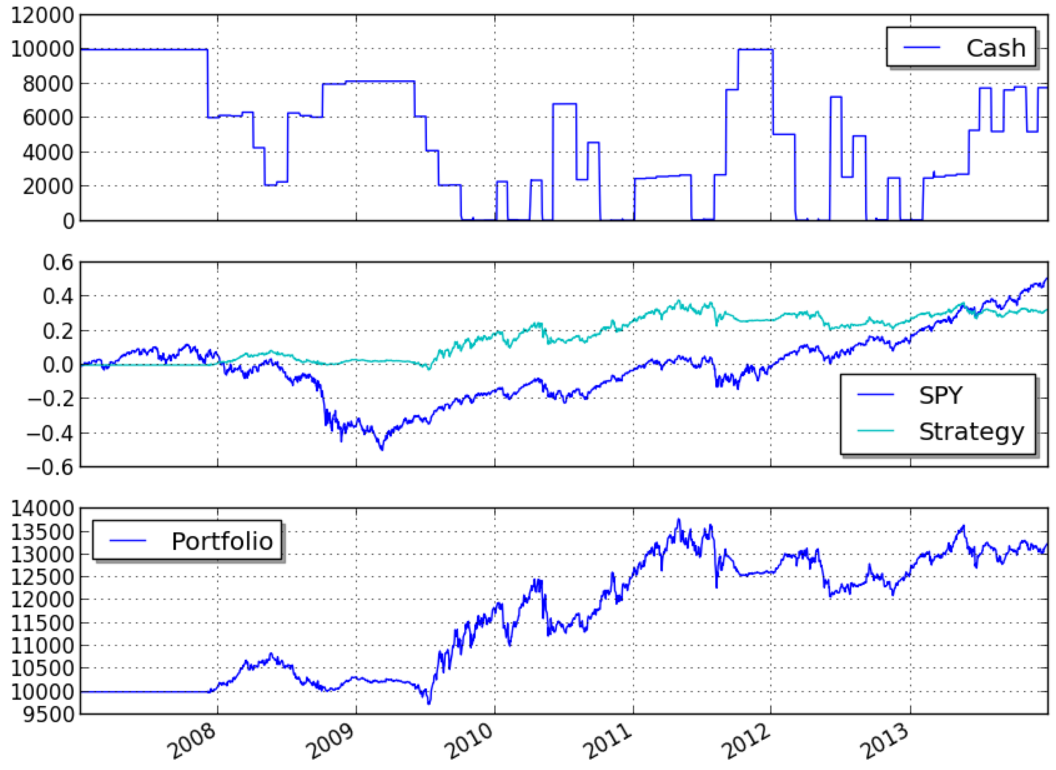
```
from pyalgotrade import strategy
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.technical import ma
from pyalgotrade.technical import cumret
from pyalgotrade.stratanalyzer import sharpe
from pyalgotrade.stratanalyzer import returns


class MarketTiming(strategy.BacktestingStrategy):
    def __init__(self, feed, instrumentsByClass, initialCash):
        strategy.BacktestingStrategy.__init__(self, feed, initialCash)
        self.setUseAdjustedValues(True)
        self.__instrumentsByClass = instrumentsByClass
        self.__rebalanceMonth = None
        self.__sharesToBuy = {}
        # Initialize indicators for each instrument.
        self.__sma = {}
        for assetClass in instrumentsByClass:
            for instrument in instrumentsByClass[assetClass]:
                priceDS = feed[instrument].getPriceDataSeries()
                self.__sma[instrument] = ma.SMA(priceDS, 200)

    def _shouldRebalance(self, dateTime):
        return dateTime.month != self.__rebalanceMonth

    def _getRank(self, instrument):
        # If the price is below the SMA, then this instrument doesn't rank at
        # all.
        smas = self.__sma[instrument]
        price = self.getLastPrice(instrument)
        if len(smas) == 0 or smas[-1] is None or price < smas[-1]:
            return None

        # Rank based on 20 day returns.
        ret = None
        lookBack = 20
        priceDS = self.getFeed()[instrument].getPriceDataSeries()
        if len(priceDS) >= lookBack and smas[-1] is not None and smas[-1*lookBack] is␣
↪not None:
            ret = (priceDS[-1] - priceDS[-1*lookBack]) / float(priceDS[-1*lookBack])
        return ret

    def _getTopByClass(self, assetClass):
        # Find the instrument with the highest rank.
        ret = None
        highestRank = None
        for instrument in self.__instrumentsByClass[assetClass]:
            rank = self._getRank(instrument)
            if rank is not None and (highestRank is None or rank > highestRank):
                highestRank = rank
                ret = instrument
        return ret

    def _getTop(self):
        ret = {}
        for assetClass in self.__instrumentsByClass:
            ret[assetClass] = self._getTopByClass(assetClass)
        return ret

    def _placePendingOrders(self):
        remainingCash = self.getBroker().getCash() * 0.9  # Use less chash just in␣
↪case price changes too much.

        for instrument in self.__sharesToBuy:
            orderSize = self.__sharesToBuy[instrument]
            if orderSize > 0:
                # Adjust the order size based on available cash.
                lastPrice = self.getLastPrice(instrument)
                cost = orderSize * lastPrice
                while cost > remainingCash and orderSize > 0:
                    orderSize -= 1
                    cost = orderSize * lastPrice
                if orderSize > 0:
                    remainingCash -= cost
                    assert(remainingCash >= 0)

            if orderSize != 0:
                self.info("Placing market order for %d %s shares" % (orderSize,␣
↪instrument))
                self.marketOrder(instrument, orderSize, goodTillCanceled=True)
                self.__sharesToBuy[instrument] -= orderSize

    def _logPosSize(self):
        totalEquity = self.getBroker().getEquity()
        positions = self.getBroker().getPositions()
        for instrument in self.getBroker().getPositions():
            posSize = positions[instrument] * self.getLastPrice(instrument) /␣
↪totalEquity * 100
            self.info("%s - %0.2f %%" % (instrument, posSize))

    def _rebalance(self):
        self.info("Rebalancing")

        # Cancel all active/pending orders.
        for order in self.getBroker().getActiveOrders():
            self.getBroker().cancelOrder(order)

        cashPerAssetClass = self.getBroker().getEquity() / float(len(self.__
↪instrumentsByClass))
        self.__sharesToBuy = {}

        # Calculate which positions should be open during the next period.
        topByClass = self._getTop()
        for assetClass in topByClass:
            instrument = topByClass[assetClass]
            self.info("Best for class %s: %s" % (assetClass, instrument))
            if instrument is not None:
                lastPrice = self.getLastPrice(instrument)
                cashForInstrument = cashPerAssetClass - self.getBroker().
↪getShares(instrument) * lastPrice
                # This may yield a negative value and we have to reduce this
                # position.
                self.__sharesToBuy[instrument] = int(cashForInstrument / lastPrice)

        # Calculate which positions should be closed.
        for instrument in self.getBroker().getPositions():
            if instrument not in list(topByClass.values()):
                currentShares = self.getBroker().getShares(instrument)
                assert(instrument not in self.__sharesToBuy)
                self.__sharesToBuy[instrument] = currentShares * -1

    def getSMA(self, instrument):
        return self.__sma[instrument]

    def onBars(self, bars):
        currentDateTime = bars.getDateTime()

        if self._shouldRebalance(currentDateTime):
            self.__rebalanceMonth = currentDateTime.month
            self._rebalance()

        self._placePendingOrders()


def main(plot):
    initialCash = 10000
    instrumentsByClass = {
        "US Stocks": ["VTI"],
        "Foreign Stocks": ["VEU"],
        "US 10 Year Government Bonds": ["IEF"],
        "Real Estate": ["VNQ"],
        "Commodities": ["DBC"],
    }

    # Download the bars.
    instruments = ["SPY"]
    for assetClass in instrumentsByClass:
        instruments.extend(instrumentsByClass[assetClass])
    feed = yahoofinance.build_feed(instruments, 2007, 2013, "data", skipErrors=True)

    strat = MarketTiming(feed, instrumentsByClass, initialCash)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)
    returnsAnalyzer = returns.Returns()
    strat.attachAnalyzer(returnsAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, False, False, True)
        plt.getOrCreateSubplot("cash").addCallback("Cash", lambda x: strat.
↪getBroker().getCash())
        # Plot strategy vs. SPY cumulative returns.
        plt.getOrCreateSubplot("returns").addDataSeries("SPY", cumret.
↪CumulativeReturn(feed["SPY"].getPriceDataSeries()))
        plt.getOrCreateSubplot("returns").addDataSeries("Strategy", returnsAnalyzer.
↪getCumulativeReturns())

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))
    print("Returns: %.2f %%" % (returnsAnalyzer.getCumulativeReturns()[-1] * 100))

    if plot:
        plt.plot()
```

```
if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2014-05-25 22:36:59,740 yahoofinance [INFO] Creating data directory
2014-05-25 22:36:59,740 yahoofinance [INFO] Downloading SPY 2007 to data/SPY-2007-
↪yahoofinance.csv
2014-05-25 22:37:06,332 yahoofinance [INFO] Downloading VTI 2007 to data/VTI-2007-
↪yahoofinance.csv
2014-05-25 22:37:10,666 yahoofinance [INFO] Downloading DBC 2007 to data/DBC-2007-
↪yahoofinance.csv
2014-05-25 22:37:13,102 yahoofinance [INFO] Downloading IEF 2007 to data/IEF-2007-
↪yahoofinance.csv
2014-05-25 22:37:19,394 yahoofinance [INFO] Downloading VEU 2007 to data/VEU-2007-
↪yahoofinance.csv
2014-05-25 22:37:27,378 yahoofinance [INFO] Downloading VNQ 2007 to data/VNQ-2007-
↪yahoofinance.csv
2014-05-25 22:37:32,771 yahoofinance [INFO] Downloading SPY 2008 to data/SPY-2008-
↪yahoofinance.csv
2014-05-25 22:37:38,326 yahoofinance [INFO] Downloading VTI 2008 to data/VTI-2008-
↪yahoofinance.csv
2014-05-25 22:37:42,262 yahoofinance [INFO] Downloading DBC 2008 to data/DBC-2008-
↪yahoofinance.csv
.
.
.
2013-12-02 00:00:00 strategy [INFO] Rebalancing
2013-12-02 00:00:00 strategy [INFO] Best for class US Stocks: VTI
2013-12-02 00:00:00 strategy [INFO] Best for class Commodities: None
2013-12-02 00:00:00 strategy [INFO] Best for class US 10 Year Government Bonds: None
2013-12-02 00:00:00 strategy [INFO] Best for class Foreign Stocks: VEU
2013-12-02 00:00:00 strategy [INFO] Best for class Real Estate: None
2013-12-02 00:00:00 strategy [INFO] Placing market order for -1 VTI shares
2013-12-02 00:00:00 strategy [INFO] Placing market order for -39 VNQ shares
Sharpe ratio: -0.06
Returns: 32.97 %
```

最终结果绘制如下图所示:



均值回归

Ernie Chan's 黄金 vs. 黄金矿工

本例基于:

- http://epchan.blogspot.com.ar/2006/11/gold-vs-gold-miners-another-arbitrage.html
- https://www.quantopian.com/posts/ernie-chans-gold-vs-gold-miners-stat-arb

```
from pyalgotrade import strategy
from pyalgotrade import dataseries
from pyalgotrade.dataseries import aligned
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.stratanalyzer import sharpe

import numpy as np
import statsmodels.api as sm
```

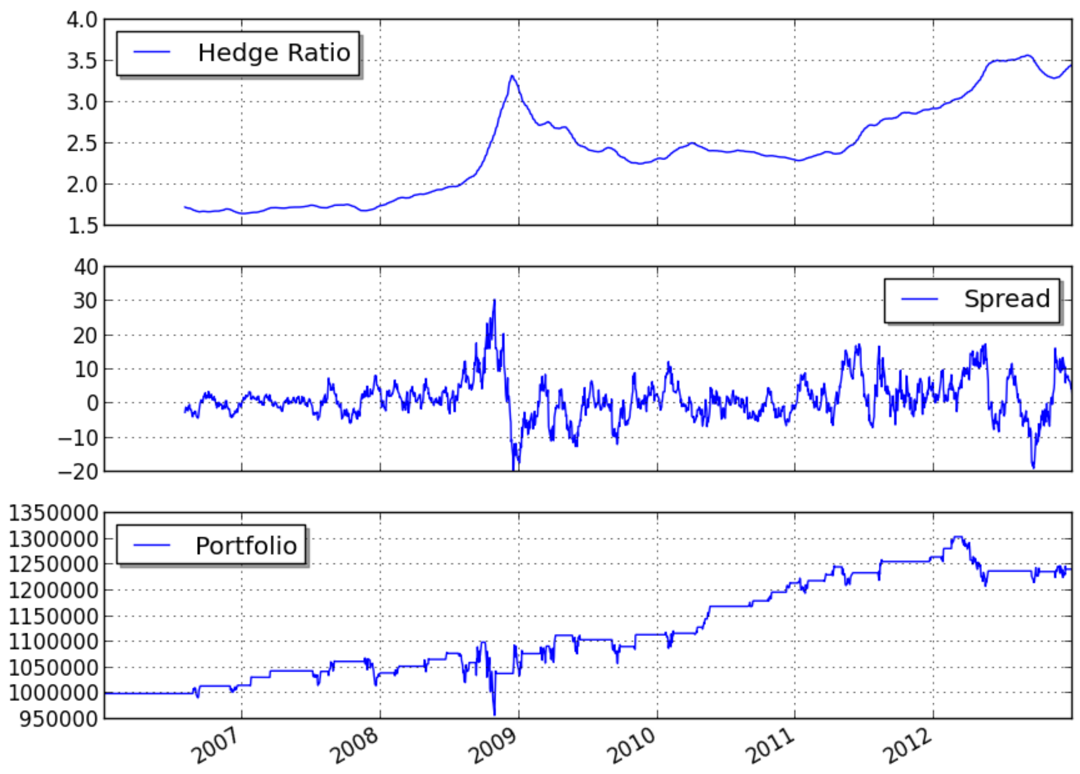
```
def get_beta(values1, values2):
    # http://statsmodels.sourceforge.net/stable/regression.html
    model = sm.OLS(values1, values2)
    results = model.fit()
    return results.params[0]


class StatArbHelper:
    def __init__(self, ds1, ds2, windowSize):
        # We're going to use datetime aligned versions of the dataseries.
        self.__ds1, self.__ds2 = aligned.datetime_aligned(ds1, ds2)
        self.__windowSize = windowSize
        self.__hedgeRatio = None
        self.__spread = None
        self.__spreadMean = None
        self.__spreadStd = None
        self.__zScore = None

    def getSpread(self):
        return self.__spread

    def getSpreadMean(self):
        return self.__spreadMean

    def getSpreadStd(self):
        return self.__spreadStd

    def getZScore(self):
        return self.__zScore

    def getHedgeRatio(self):
        return self.__hedgeRatio

    def __updateHedgeRatio(self, values1, values2):
        self.__hedgeRatio = get_beta(values1, values2)

    def __updateSpreadMeanAndStd(self, values1, values2):
        if self.__hedgeRatio is not None:
            spread = values1 - values2 * self.__hedgeRatio
            self.__spreadMean = spread.mean()
            self.__spreadStd = spread.std(ddof=1)

    def __updateSpread(self):
        if self.__hedgeRatio is not None:
            self.__spread = self.__ds1[-1] - self.__hedgeRatio * self.__ds2[-1]

    def __updateZScore(self):
        if self.__spread is not None and self.__spreadMean is not None and self.__
↪spreadStd is not None:
            self.__zScore = (self.__spread - self.__spreadMean) / float(self.__
↪spreadStd)

    def update(self):
        if len(self.__ds1) >= self.__windowSize:
            values1 = np.asarray(self.__ds1[-1*self.__windowSize:])
            values2 = np.asarray(self.__ds2[-1*self.__windowSize:])
            self.__updateHedgeRatio(values1, values2)
            self.__updateSpread()
            self.__updateSpreadMeanAndStd(values1, values2)
            self.__updateZScore()


class StatArb(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument1, instrument2, windowSize):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.setUseAdjustedValues(True)
        self.__statArbHelper = StatArbHelper(feed[instrument1].
↪getAdjCloseDataSeries(), feed[instrument2].getAdjCloseDataSeries(), windowSize)
        self.__i1 = instrument1
        self.__i2 = instrument2

        # These are used only for plotting purposes.
        self.__spread = dataseries.SequenceDataSeries()
        self.__hedgeRatio = dataseries.SequenceDataSeries()

    def getSpreadDS(self):
        return self.__spread

    def getHedgeRatioDS(self):
        return self.__hedgeRatio

    def __getOrderSize(self, bars, hedgeRatio):
        cash = self.getBroker().getCash(False)
        price1 = bars[self.__i1].getAdjClose()
        price2 = bars[self.__i2].getAdjClose()
        size1 = int(cash / (price1 + hedgeRatio * price2))
        size2 = int(size1 * hedgeRatio)
        return (size1, size2)

    def buySpread(self, bars, hedgeRatio):
        amount1, amount2 = self.__getOrderSize(bars, hedgeRatio)
        self.marketOrder(self.__i1, amount1)
        self.marketOrder(self.__i2, amount2 * -1)

    def sellSpread(self, bars, hedgeRatio):
        amount1, amount2 = self.__getOrderSize(bars, hedgeRatio)
        self.marketOrder(self.__i1, amount1 * -1)
        self.marketOrder(self.__i2, amount2)

    def reducePosition(self, instrument):
        currentPos = self.getBroker().getShares(instrument)
        if currentPos > 0:
            self.marketOrder(instrument, currentPos * -1)
        elif currentPos < 0:
            self.marketOrder(instrument, currentPos * -1)

    def onBars(self, bars):
        self.__statArbHelper.update()

        # These is used only for plotting purposes.
        self.__spread.appendWithDateTime(bars.getDateTime(), self.__statArbHelper.
↪getSpread())
        self.__hedgeRatio.appendWithDateTime(bars.getDateTime(), self.__statArbHelper.
↪getHedgeRatio())

        if bars.getBar(self.__i1) and bars.getBar(self.__i2):
            hedgeRatio = self.__statArbHelper.getHedgeRatio()
            zScore = self.__statArbHelper.getZScore()
            if zScore is not None:
                currentPos = abs(self.getBroker().getShares(self.__i1)) + abs(self.
↪getBroker().getShares(self.__i2))
                if abs(zScore) <= 1 and currentPos != 0:
                    self.reducePosition(self.__i1)
                    self.reducePosition(self.__i2)
                elif zScore <= -2 and currentPos == 0:  # Buy spread when its value␣
↪drops below 2 standard deviations.
                    self.buySpread(bars, hedgeRatio)
                elif zScore >= 2 and currentPos == 0:  # Short spread when its value␣
↪rises above 2 standard deviations.
                    self.sellSpread(bars, hedgeRatio)


def main(plot):
    instruments = ["gld", "gdx"]
    windowSize = 50

    # Download the bars.
    feed = yahoofinance.build_feed(instruments, 2006, 2012, ".")

    strat = StatArb(feed, instruments[0], instruments[1], windowSize)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, False, False, True)
        plt.getOrCreateSubplot("hedge").addDataSeries("Hedge Ratio", strat.
↪getHedgeRatioDS())
        plt.getOrCreateSubplot("spread").addDataSeries("Spread", strat.getSpreadDS())

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2013-09-21 00:02:35,136 yahoofinance [INFO] Creating data directory
2013-09-21 00:02:35,136 yahoofinance [INFO] Downloading gld 2006 to data/gld-2006-
↪yahoofinance.csv
2013-09-21 00:02:35,899 yahoofinance [INFO] Downloading gdx 2006 to data/gdx-2006-
↪yahoofinance.csv
2013-09-21 00:02:36,637 yahoofinance [INFO] Downloading gld 2007 to data/gld-2007-
↪yahoofinance.csv
2013-09-21 00:02:37,265 yahoofinance [INFO] Downloading gdx 2007 to data/gdx-2007-
↪yahoofinance.csv
2013-09-21 00:02:37,881 yahoofinance [INFO] Downloading gld 2008 to data/gld-2008-
↪yahoofinance.csv
2013-09-21 00:02:38,462 yahoofinance [INFO] Downloading gdx 2008 to data/gdx-2008-
↪yahoofinance.csv
```

```
2013-09-21 00:02:39,243 yahoofinance [INFO] Downloading gld 2009 to data/gld-2009-
↪yahoofinance.csv
2013-09-21 00:02:39,996 yahoofinance [INFO] Downloading gdx 2009 to data/gdx-2009-
↪yahoofinance.csv
2013-09-21 00:02:40,577 yahoofinance [INFO] Downloading gld 2010 to data/gld-2010-
↪yahoofinance.csv
2013-09-21 00:02:42,630 yahoofinance [INFO] Downloading gdx 2010 to data/gdx-2010-
↪yahoofinance.csv
2013-09-21 00:02:43,397 yahoofinance [INFO] Downloading gld 2011 to data/gld-2011-
↪yahoofinance.csv
2013-09-21 00:02:44,153 yahoofinance [INFO] Downloading gdx 2011 to data/gdx-2011-
↪yahoofinance.csv
2013-09-21 00:02:44,901 yahoofinance [INFO] Downloading gld 2012 to data/gld-2012-
↪yahoofinance.csv
2013-09-21 00:02:45,965 yahoofinance [INFO] Downloading gdx 2012 to data/gdx-2012-
↪yahoofinance.csv
Sharpe ratio: -0.20
```

最终结果绘制如下图所示:



You can get better returns by tunning the window size as well as the entry and exit values for the z-score.

布林带

This example is based on:

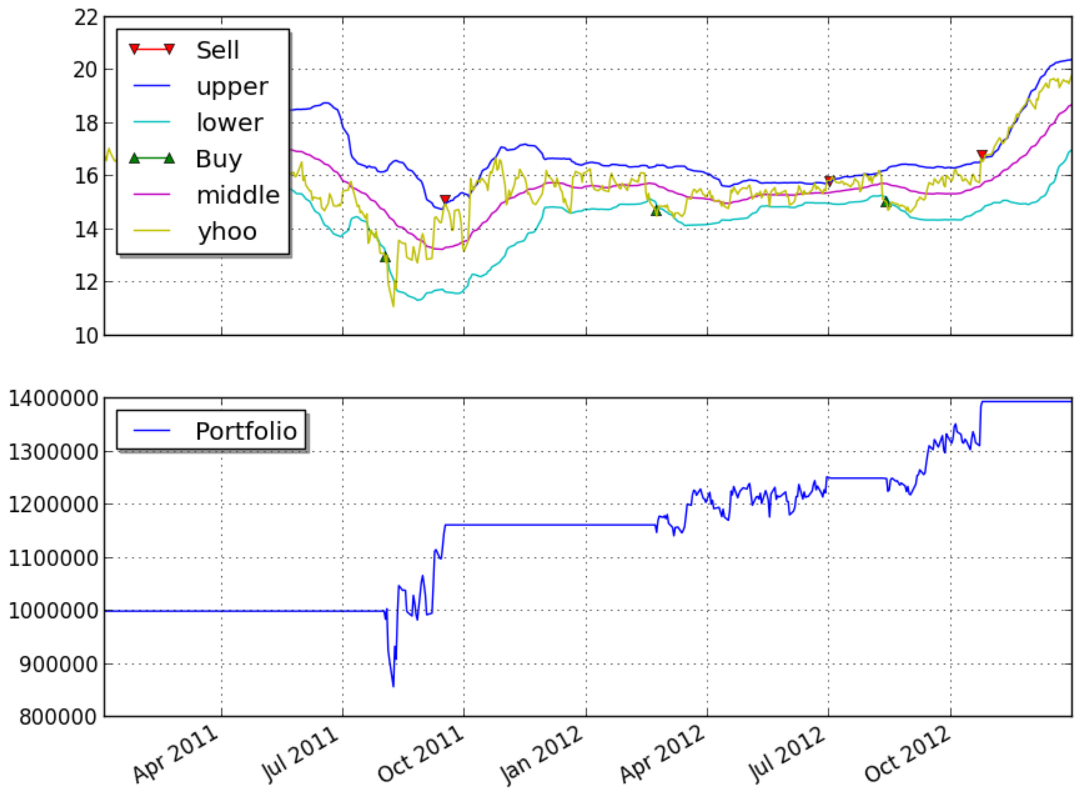
- http://www.investopedia.com/articles/trading/07/bollinger.asp

```
from pyalgotrade import strategy
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.technical import bollinger
from pyalgotrade.stratanalyzer import sharpe


class BBands(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, bBandsPeriod):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        self.__bbands = bollinger.BollingerBands(feed[instrument].
↪getCloseDataSeries(), bBandsPeriod, 2)

    def getBollingerBands(self):
        return self.__bbands

    def onBars(self, bars):
        lower = self.__bbands.getLowerBand()[-1]
        upper = self.__bbands.getUpperBand()[-1]
        if lower is None:
            return

        shares = self.getBroker().getShares(self.__instrument)
        bar = bars[self.__instrument]
        if shares == 0 and bar.getClose() < lower:
            sharesToBuy = int(self.getBroker().getCash(False) / bar.getClose())
            self.marketOrder(self.__instrument, sharesToBuy)
        elif shares > 0 and bar.getClose() > upper:
            self.marketOrder(self.__instrument, -1*shares)


def main(plot):
    instrument = "yhoo"
    bBandsPeriod = 40

    # Download the bars.
    feed = yahoofinance.build_feed([instrument], 2011, 2012, ".")

    strat = BBands(feed, instrument, bBandsPeriod)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, True, True, True)
        plt.getInstrumentSubplot(instrument).addDataSeries("upper", strat.
↪getBollingerBands().getUpperBand())
        plt.getInstrumentSubplot(instrument).addDataSeries("middle", strat.
↪getBollingerBands().getMiddleBand())
        plt.getInstrumentSubplot(instrument).addDataSeries("lower", strat.
↪getBollingerBands().getLowerBand())

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2013-09-21 00:06:07,740 yahoofinance [INFO] Creating data directory
2013-09-21 00:06:07,741 yahoofinance [INFO] Downloading yhoo 2011 to data/yhoo-2011-
↪yahoofinance.csv
2013-09-21 00:06:09,621 yahoofinance [INFO] Downloading yhoo 2012 to data/yhoo-2012-
↪yahoofinance.csv
Sharpe ratio: 0.71
```

最终结果绘制如下图所示:



You can get better returns by tunning the Bollinger Bands period as well as the entry and exit points.

RSI2

This example is based on a strategy known as RSI2 (http://stockcharts.com/school/doku.php?id=chart_school:trading_strategies:rsi2) which requires the following parameters:

- An SMA period for trend identification. We'll call this entrySMA.
- A smaller SMA period for the exit point. We'll call this exitSMA.
- An RSI period for entering both short/long positions. We'll call this rsiPeriod.
- An RSI oversold threshold for long position entry. We'll call this overSoldThreshold.
- An RSI overbought threshold for short position entry. We'll call this overBoughtThreshold.

Save this code as rsi2.py:

```python
from pyalgotrade import strategy
from pyalgotrade.technical import ma
from pyalgotrade.technical import rsi
from pyalgotrade.technical import cross


class RSI2(strategy.BacktestingStrategy):
    def __init__(self, feed, instrument, entrySMA, exitSMA, rsiPeriod,␣
↪overBoughtThreshold, overSoldThreshold):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.__instrument = instrument
        # We'll use adjusted close values, if available, instead of regular close␣
↪values.
        if feed.barsHaveAdjClose():
            self.setUseAdjustedValues(True)
        self.__priceDS = feed[instrument].getPriceDataSeries()
        self.__entrySMA = ma.SMA(self.__priceDS, entrySMA)
        self.__exitSMA = ma.SMA(self.__priceDS, exitSMA)
        self.__rsi = rsi.RSI(self.__priceDS, rsiPeriod)
        self.__overBoughtThreshold = overBoughtThreshold
        self.__overSoldThreshold = overSoldThreshold
        self.__longPos = None
        self.__shortPos = None

    def getEntrySMA(self):
        return self.__entrySMA

    def getExitSMA(self):
        return self.__exitSMA

    def getRSI(self):
        return self.__rsi

    def onEnterCanceled(self, position):
        if self.__longPos == position:
            self.__longPos = None
        elif self.__shortPos == position:
            self.__shortPos = None
        else:
            assert(False)

    def onExitOk(self, position):
        if self.__longPos == position:
            self.__longPos = None
        elif self.__shortPos == position:
            self.__shortPos = None
        else:
            assert(False)

    def onExitCanceled(self, position):
        # If the exit was canceled, re-submit it.
        position.exitMarket()

    def onBars(self, bars):
        # Wait for enough bars to be available to calculate SMA and RSI.
        if self.__exitSMA[-1] is None or self.__entrySMA[-1] is None or self.__rsi[-
↪1] is None:
            return

        bar = bars[self.__instrument]
        if self.__longPos is not None:
            if self.exitLongSignal():
                self.__longPos.exitMarket()
        elif self.__shortPos is not None:
            if self.exitShortSignal():
                self.__shortPos.exitMarket()
        else:
            if self.enterLongSignal(bar):
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                self.__longPos = self.enterLong(self.__instrument, shares, True)
            elif self.enterShortSignal(bar):
                shares = int(self.getBroker().getCash() * 0.9 / bars[self.__
↪instrument].getPrice())
                self.__shortPos = self.enterShort(self.__instrument, shares, True)

    def enterLongSignal(self, bar):
        return bar.getPrice() > self.__entrySMA[-1] and self.__rsi[-1] <= self.__
↪overSoldThreshold

    def exitLongSignal(self):
        return cross.cross_above(self.__priceDS, self.__exitSMA) and not self.__
↪longPos.exitActive()

    def enterShortSignal(self, bar):
        return bar.getPrice() < self.__entrySMA[-1] and self.__rsi[-1] >= self.__
↪overBoughtThreshold

    def exitShortSignal(self):
        return cross.cross_below(self.__priceDS, self.__exitSMA) and not self.__
↪shortPos.exitActive()
```

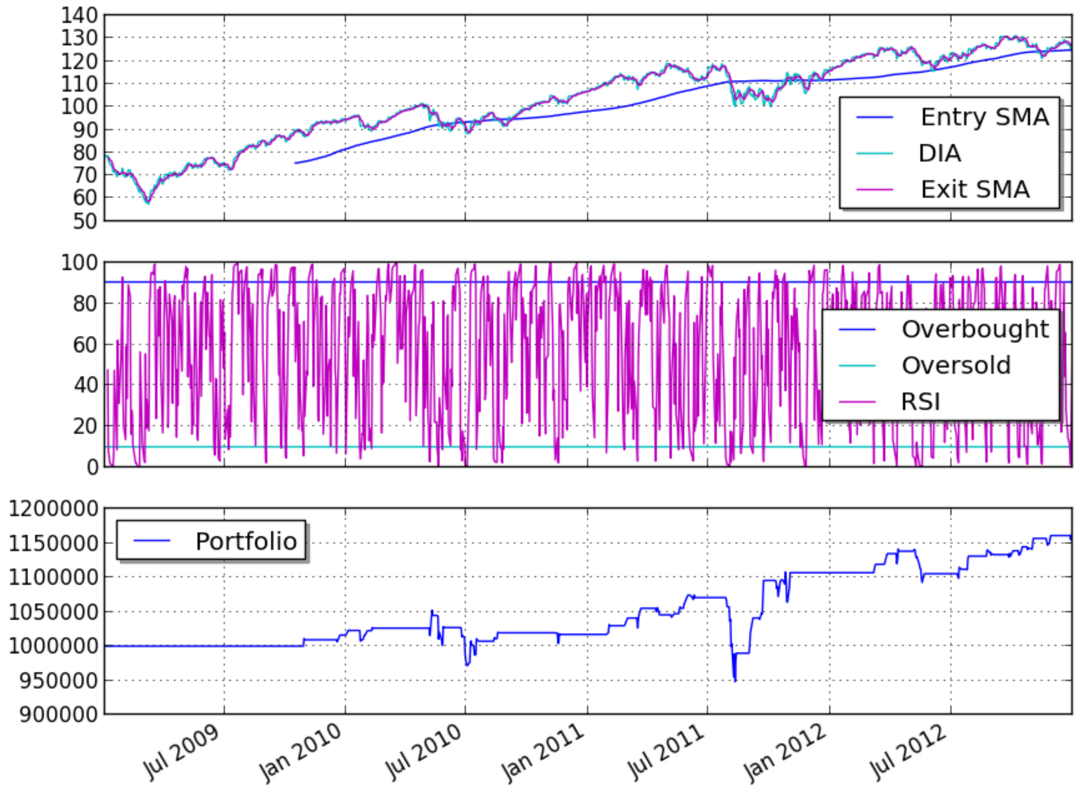
and use the following code to execute the strategy:

```
import rsi2
from pyalgotrade import plotter
from pyalgotrade.tools import yahoofinance
from pyalgotrade.stratanalyzer import sharpe


def main(plot):
    instrument = "DIA"
    entrySMA = 200
    exitSMA = 5
    rsiPeriod = 2
    overBoughtThreshold = 90
    overSoldThreshold = 10

    # Download the bars.
    feed = yahoofinance.build_feed([instrument], 2009, 2012, ".")

    strat = rsi2.RSI2(feed, instrument, entrySMA, exitSMA, rsiPeriod,␣
↪overBoughtThreshold, overSoldThreshold)
    sharpeRatioAnalyzer = sharpe.SharpeRatio()
    strat.attachAnalyzer(sharpeRatioAnalyzer)

    if plot:
        plt = plotter.StrategyPlotter(strat, True, False, True)
        plt.getInstrumentSubplot(instrument).addDataSeries("Entry SMA", strat.
↪getEntrySMA())
        plt.getInstrumentSubplot(instrument).addDataSeries("Exit SMA", strat.
↪getExitSMA())
        plt.getOrCreateSubplot("rsi").addDataSeries("RSI", strat.getRSI())
        plt.getOrCreateSubplot("rsi").addLine("Overbought", overBoughtThreshold)
        plt.getOrCreateSubplot("rsi").addLine("Oversold", overSoldThreshold)

    strat.run()
    print("Sharpe ratio: %.2f" % sharpeRatioAnalyzer.getSharpeRatio(0.05))

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2014-05-03 13:49:35,354 yahoofinance [INFO] Downloading DIA 2009 to ./DIA-2009-
↪yahoofinance.csv
2014-05-03 13:49:36,388 yahoofinance [INFO] Downloading DIA 2010 to ./DIA-2010-
↪yahoofinance.csv
2014-05-03 13:49:36,900 yahoofinance [INFO] Downloading DIA 2011 to ./DIA-2011-
↪yahoofinance.csv
2014-05-03 13:49:37,457 yahoofinance [INFO] Downloading DIA 2012 to ./DIA-2012-
↪yahoofinance.csv
Sharpe ratio: -0.11
```

最终结果绘制如下图所示:



You can get better returns by tunning the different parameters.

其他

Quandl 合集

The purpose of this example is to show how to integrate price data along with any time-series data in CSV format from Quandl into a strategy.

We'll use the following CSV data from Quandl: http://www.quandl.com/OFDP-Open-Financial-Data-Project/GOLD_2-LBMA-Gold-Price-London-Fixings-P-M

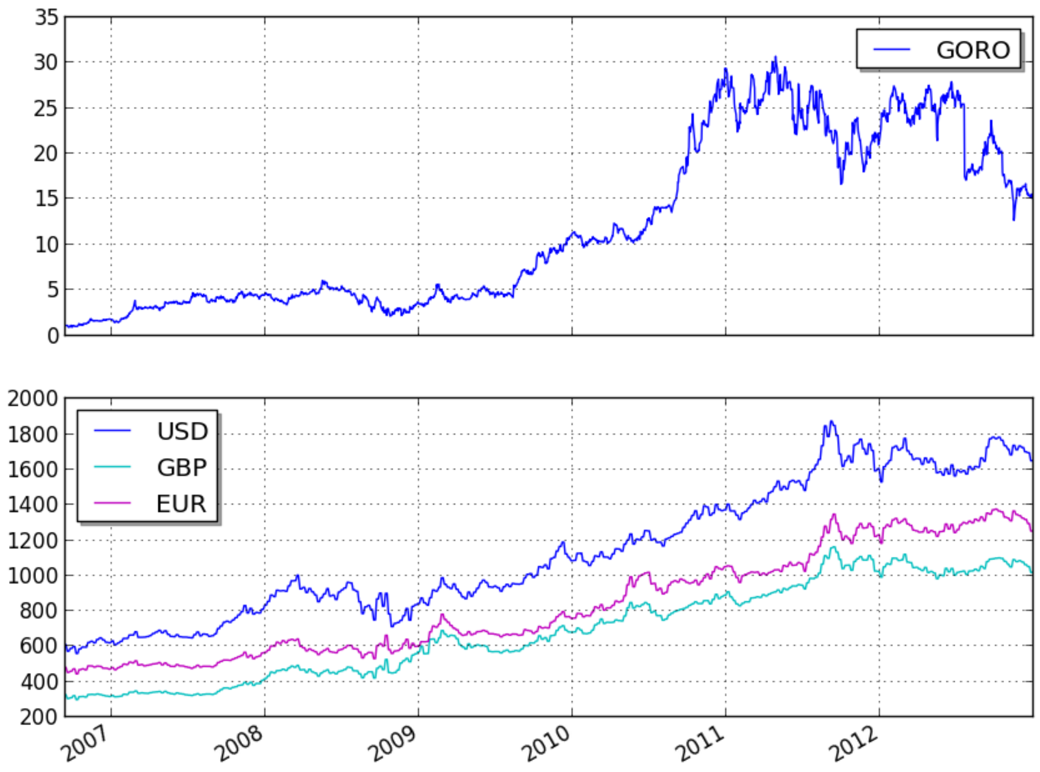
```
from pyalgotrade import strategy
from pyalgotrade import plotter
from pyalgotrade.tools import quandl
from pyalgotrade.feed import csvfeed
import datetime


class MyStrategy(strategy.BacktestingStrategy):
    def __init__(self, feed, quandlFeed, instrument):
        strategy.BacktestingStrategy.__init__(self, feed)
        self.setUseAdjustedValues(True)
        self.__instrument = instrument

        # It is VERY important to add the the extra feed to the event dispatch loop␣
↪before
        # running the strategy.
        self.getDispatcher().addSubject(quandlFeed)

        # Subscribe to events from the Quandl feed.
        quandlFeed.getNewValuesEvent().subscribe(self.onQuandlData)

    def onQuandlData(self, dateTime, values):
        self.info(values)

    def onBars(self, bars):
        self.info(bars[self.__instrument].getAdjClose())


def main(plot):
    instruments = ["GORO"]

    # Download GORO bars using WIKI source code.
    feed = quandl.build_feed("WIKI", instruments, 2006, 2012, ".")

    # Load Quandl CSV downloaded from http://www.quandl.com/OFDP-Open-Financial-Data-
↪Project/GOLD_2-LBMA-Gold-Price-London-Fixings-P-M
    quandlFeed = csvfeed.Feed("Date", "%Y-%m-%d")
    quandlFeed.setDateRange(datetime.datetime(2006, 1, 1), datetime.datetime(2012, 12,
↪ 31))
    quandlFeed.addValuesFromCSV("quandl_gold_2.csv")

    myStrategy = MyStrategy(feed, quandlFeed, instruments[0])

    if plot:
        plt = plotter.StrategyPlotter(myStrategy, True, False, False)
        plt.getOrCreateSubplot("quandl").addDataSeries("USD", quandlFeed["USD"])
        plt.getOrCreateSubplot("quandl").addDataSeries("EUR", quandlFeed["EUR"])
        plt.getOrCreateSubplot("quandl").addDataSeries("GBP", quandlFeed["GBP"])

    myStrategy.run()

    if plot:
        plt.plot()


if __name__ == "__main__":
    main(True)
```

本例输出结果如下:

```
2006-01-01 00:00:00 strategy [INFO] {'USD': 513.0, 'GBP': 298.204, 'EUR': 433.533}
2006-01-08 00:00:00 strategy [INFO] {'USD': 535.25, 'GBP': 302.572, 'EUR': 440.173}
2006-01-15 00:00:00 strategy [INFO] {'USD': 548.25, 'GBP': 309.781, 'EUR': 454.489}
2006-01-22 00:00:00 strategy [INFO] {'USD': 567.25, 'GBP': 321.152, 'EUR': 468.802}
2006-01-29 00:00:00 strategy [INFO] {'USD': 561.75, 'GBP': 315.147, 'EUR': 460.526}
2006-02-05 00:00:00 strategy [INFO] {'USD': 569.0, 'GBP': 322.562, 'EUR': 474.167}
2006-02-12 00:00:00 strategy [INFO] {'USD': 557.0, 'GBP': 317.198, 'EUR': 463.78}
```

```
2006-02-19 00:00:00 strategy [INFO] {'USD': 551.7, 'GBP': 317.251, 'EUR': 463.224}
2006-02-26 00:00:00 strategy [INFO] {'USD': 554.15, 'GBP': 316.838, 'EUR': 465.555}
2006-03-05 00:00:00 strategy [INFO] {'USD': 565.0, 'GBP': 322.029, 'EUR': 469.854}
.
.
.
2012-12-19 00:00:00 strategy [INFO] 15.43
2012-12-20 00:00:00 strategy [INFO] 15.39
2012-12-21 00:00:00 strategy [INFO] 15.35
2012-12-23 00:00:00 strategy [INFO] {'USD': 1651.5, 'GBP': 1019.256, 'EUR': 1253.701}
2012-12-24 00:00:00 strategy [INFO] 15.2
2012-12-26 00:00:00 strategy [INFO] 15.56
2012-12-27 00:00:00 strategy [INFO] 15.24
2012-12-28 00:00:00 strategy [INFO] 15.09
2012-12-30 00:00:00 strategy [INFO] {'USD': 1657.5, 'GBP': 1027.206, 'EUR': 1253.024}
2012-12-31 00:00:00 strategy [INFO] 15.41
```

最终结果绘制如下图所示:



CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python 模块索引

索引