

Во время посещения сайта вы соглашаетесь с использованием файлов [cookie](#)

Хорошо



Михаил Шардин ★

личный блог



18 ноября 2024, 05:30

+ Подписаться

## Мой первый и неудачный опыт поиска торговой стратегии для Московской биржи

Когда закончил писать [механизм своего торгового робота](#) обнаружил, что самое главное всё таки не сам механизм, а стратегия, по которой этот механизм будет работать.

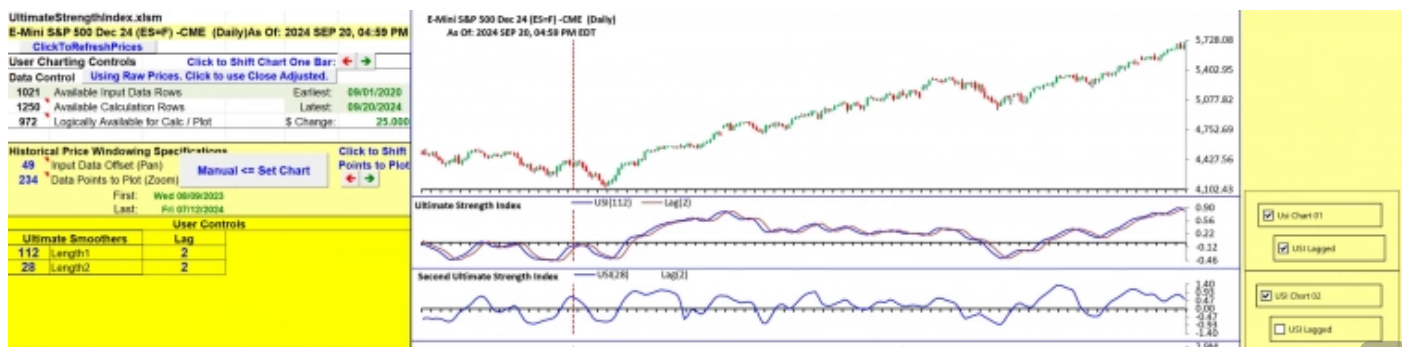
Первый тесты на истории показали что с доходностью и тем более с тем как доходность портфеля компенсирует принимаемый риск (коэффициент Шарпа) проблемы, но неудачный опыт тоже опыт, поэтому решил описать его в статье.

Первый и самый важный вопрос — при помощи чего проводить тесты торговой стратегии на исторических данных? В какой программе или при помощи какой библиотеки создавать стратегию и потом прогонять её на истории?

Раз мой торговый робот создан в среде исполнения JavaScript Node.js, то и тесты в идеале должны проводится на чём-то схожем. Но забегаю немного вперёд скажу что получилось по другому.

### Windows? macOS? Linux?

Раз сам механизм робота кросс-платформенный, то хотелось чтобы и тесты можно было проводить при помощи кросс-платформенной утилиты. Однако когда рассматривал самые популярные программы, то обнаружилось что все программы из списка только для Windows. Кроме TradingView, который является веб-сервисом и Excel — который есть и для macOS.



Откройте счёт в ВТБ Мои Инвестиции

Введите текст комментария

построения графиков, автоматизации стратегий и бэктестинга для акций, опционов, фьючерсов и криптовалют.

- **NinjaTrader**: торговое программное обеспечение для фьючерсов и форекс; отлично подходит для расширенного построения графиков, бэктестинга и автоматизированной торговли.
- **MetaStock**: фокусируется на техническом анализе и бэктестинге с обширными инструментами для построения графиков и индикаторов, популярен среди трейдеров акциями.
- **Wealth-Lab**: платформа, известная расширенным бэктестингом и разработкой торговых стратегий с мощной поддержкой портфелей из нескольких активов.
- **TradingView**: удобная в использовании платформа для построения графиков с социальными функциями; отлично подходит для технического анализа, обмена идеями и базового бэктестинга стратегий.
- **RealTest**: легкое программное обеспечение для бэктестинга и разработки стратегий, известное своей скоростью и простотой, ориентированное на системных трейдеров.
- **Neuroshell Trader**: специализируется на прогнозном моделировании и анализе на основе нейронных сетей; идеально подходит для трейдеров, интересующихся машинным обучением.
- **TSLab**: платформа позволяет разрабатывать, тестировать и оптимизировать торговые системы без необходимости глубокого знания программирования.
- **The Zorro Project**: бесплатная, легкая и скриптовая платформа, предназначенная для автоматизированной торговли, бэктестинга и исследований, популярная среди алгоритмических трейдеров.
- и даже **Microsoft Excel**: универсальный инструмент для работы с электронными таблицами, часто используемый для анализа портфеля, пользовательского бэктестинга и организации данных в торговле.

Ни один из этих вариантов мне не приглянулся из-за отсутствия кросс-платформенности или этот вариант был Экселем.

## Node.js библиотеки — не смог ❌

После этого стал смотреть библиотеки для Node.js. Выбор оказался небольшой и более-менее живыми мне показались:

**grademark**: [github.com/Grademark/grademark](https://github.com/Grademark/grademark)

Библиотека Node.js для бэктестинга торговых стратегий на исторических данных

Откройте счёт в ВТБ Мои Инвестиции

## CCXT — Cryptocurrency exchange Trading Library: [github.com/ccxt/ccxt](https://github.com/ccxt/ccxt)

Библиотека Node.js для торговли криптовалютой, которая предоставляет унифицированный API для подключения и торговли на нескольких криптовалютных биржах, поддерживая как торговлю в реальном времени, так и доступ к историческим данным.

```

src > old_node15_not work > .\ backtest_grademark_not work.js > runBacktest > strategy
32 function aggregateData(minuteData, interval) {
35   minuteData.forEach((entry, index) => {
45     });
46     aggregated.push(aggCandle);
47     temp = [];
48   });
49 });
50 return aggregated;
51 }
52
53 // Функция для запуска стратегии
54 async function runBacktest(startMonth, testMonth) {
55   let strategy = grademark({
56     buy: ({ fiveMinuteCandle, hourlyCandle }) => {
57       return {
58         fiveMinuteCandle.close > calculateSMA(fiveMinuteCandle, 5) &&
59         hourlyCandle.close > calculateSMA(hourlyCandle, 60)
60       };
61     },
62     sell: ({ price, maxPrice }) => {
63       return price < maxPrice * (1 - trailingStopPercent / 100);
64     },
65   });
66
67   // Сначала оптимизируем стратегию на данных за месяц
68   let januaryData = await readCsvData('data/e6123145-9665-43e0-8413-cd61b8aa9b13_2024${startMonth}.csv');
69   let fiveMinuteCandles = aggregateData(januaryData, 5);
70   let hourlyCandles = aggregateData(januaryData, 60);
71
72   strategy.optimize({ fiveMinuteCandles, hourlyCandles });
73
74   // Далее, проводим тестирование на следующем месяце (например, февраль)
75   let februaryData = await readCsvData('data/e6123145-9665-43e0-8413-cd61b8aa9b13_2024${testMonth}.csv');
76   let testFiveMinuteCandles = aggregateData(februaryData, 5);
77   let testHourlyCandles = aggregateData(februaryData, 60);

```

Для Grademark набросал через ChatGPT конкретный пример использования:

```

const fs = require('fs');
const csvParser = require('csv-parser');
const grademark = require('grademark');

// Параметры стратегии
let trailingStopPercent = 1; // Процент падения для трейлинг-стопа
let buyThreshold = 1; // Минимальный процент для покупки

// Функция для чтения данных из CSV-файла
function readCsvData(filePath) {
  return new Promise((resolve, reject) => {
    let data = [];
    fs.createReadStream(filePath)

```

Откройте счёт в ВТБ Мои Инвестиции

```
        open: parseFloat(row[2]),
        high: parseFloat(row[3]),
        low: parseFloat(row[4]),
        close: parseFloat(row[5]),
        volume: parseInt(row[6])
    });
    })
    .on('end', () => resolve(data))
    .on('error', reject);
});
}

// Функция для агрегирования минутных данных в 5-минутные и часовые свечи
function aggregateData(minuteData, interval) {
    const aggregated = [];
    let temp = [];
    minuteData.forEach((entry, index) => {
        temp.push(entry);
        if ((index + 1) % interval === 0) {
            const aggCandle = {
                open: temp[0].open,
                high: Math.max(...temp.map(t => t.high)),
                low: Math.min(...temp.map(t => t.low)),
                close: temp[temp.length - 1].close,
                volume: temp.reduce((acc, t) => acc + t.volume, 0),
                time: temp[temp.length - 1].time
            };
            aggregated.push(aggCandle);
            temp = [];
        }
    });
    return aggregated;
}

// Функция для запуска стратегии
```

Откройте счёт в ВТБ Мои Инвестиции

```

        hourlyCandle.close > calculateSMA(hourlyCandle, 60)
    );
},
sell: ({ price, maxPrice }) => {
    return price < maxPrice * (1 - trailingStopPercent / 100);
},
});

// Сначала оптимизируем стратегию на данных за месяц
let januaryData = await readCsvData(`data/e6123145-9665-43e0-8413-cd61b8aa9b
let fiveMinuteCandles = aggregateData(januaryData, 5);
let hourlyCandles = aggregateData(januaryData, 60);

strategy.optimize({ fiveMinuteCandles, hourlyCandles });

// Далее, проводим тестирование на следующем месяце (например, февраль)
let februaryData = await readCsvData(`data/e6123145-9665-43e0-8413-cd61b8aa9
let testFiveMinuteCandles = aggregateData(februaryData, 5);
let testHourlyCandles = aggregateData(februaryData, 60);

let testResults = strategy.run({ fiveMinuteCandles: testFiveMinuteCandles, h

console.log('Результаты теста:', testResults);
}

// Пример вызова функции для тестирования
runBacktest('09', '10'); // Оптимизация на сентябрьских данных и тестирование на

// Функция для вычисления скользящей средней (SMA)
function calculateSMA(candles, period) {
    if (candles.length < period) return 0;
    const sum = candles.slice(-period).reduce((acc, candle) => acc + candle.clos
    return sum / period;
}

```

При этом криптовалюты мне не подходили. Grademark почему-то не смог установить а

Откройте счёт в ВТБ Мои Инвестиции

рассчитанных на разные уровни сложности и типы активов. Вот найденные варианты:

**Backtesting.py** [github.com/kernc/backtesting.py](https://github.com/kernc/backtesting.py)

Легкая, интуитивно понятная библиотека для векторизованного бэктестинга, включающая популярные индикаторы и метрики.

✗ 4 года не обновлялась.

**Backtrader** [github.com/mementum/backtrader](https://github.com/mementum/backtrader)

Одна из самых популярных и многофункциональных библиотек для бэктестинга.

Поддерживает несколько активов, таймфреймов, индикаторов и оптимизацию стратегий.

**PyAlgoTrade** [github.com/gbeced/pyalgotrade](https://github.com/gbeced/pyalgotrade)

Простая библиотека бэктестинга со встроенной поддержкой технических индикаторов и создания базовой стратегии.

✗ Этот репозиторий был заархивирован владельцем 13 ноября 2023 г.

**Zipline** [github.com/quantopian/zipline](https://github.com/quantopian/zipline)

Разработанная Quantopian (теперь поддерживаемая сообществом), Zipline — это надежная библиотека бэктестинга, ориентированная на событийно-управляемое бэктестирование, используемая профессионалами.

✗ 4 года не обновлялась.

**QuantConnect/Lean** [github.com/QuantConnect/Lean](https://github.com/QuantConnect/Lean)

Движок с открытым исходным кодом, лежащий в основе QuantConnect; поддерживает бэктестинг и торговлю в реальном времени для нескольких классов активов.

**VectorBT** [github.com/polakowo/vectorbt](https://github.com/polakowo/vectorbt)

Разработан для быстрого векторизованного бэктестинга и анализа стратегий непосредственно на Pandas DataFrames.

**Fastquant** [github.com/enzoampil/fastquant](https://github.com/enzoampil/fastquant)

Удобная библиотека бэктестинга, разработанная для быстрого тестирования с минимальной настройкой, вдохновленная Prophet от Facebook.

✗ 3 года не обновлялась.

**MibianLib** [github.com/yassinemaaroufi/MibianLib](https://github.com/yassinemaaroufi/MibianLib)

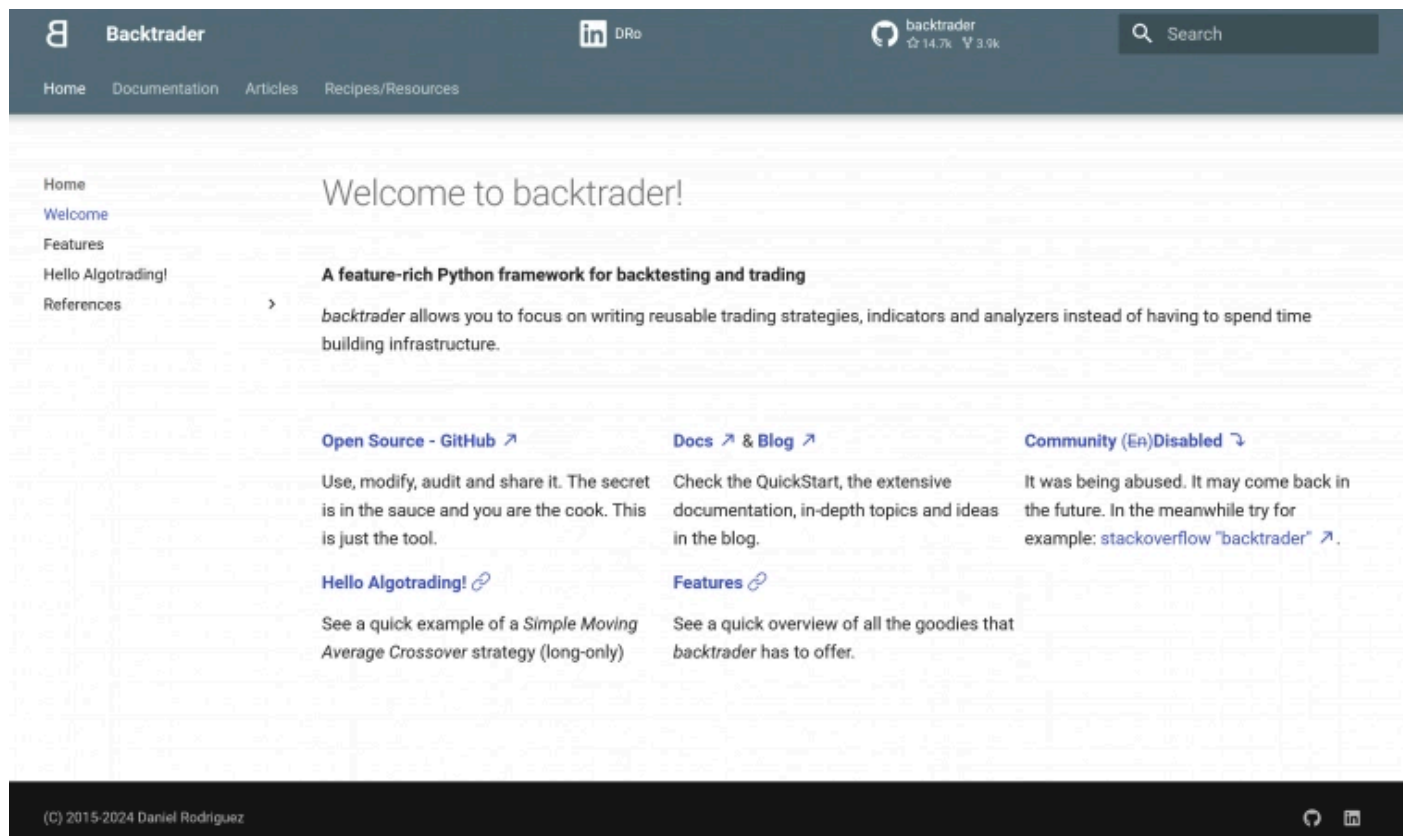
Фокусируется на ценообразовании и волатильности опционов, а не на полном бэктестинге, но полезен для стратегий, связанных с опционами.

✗ 11 лет не обновлялась.

Откройте счёт в ВТБ Мои Инвестиции

новыми версиями pandas, в частности, при повторной выборке данных для построения графиков. А новой версии Backtesting.py, которая решает эту проблему и поддерживает последние изменения API pandas просто нет.

Следующий в списке был **Backtrader** — с ним и продолжил работать.



## Идея моей торговой стратегии 💡

Хотя считается что торговая стратегия необязательно должна быть «человекочитаемой» — это вполне может быть результат обучения алгоритма, основанного на интеллектуальных технологиях (нейросети, машинное обучение и т.п.), но я решил начать с простого.

Мои условия:

1. Торговать только в лонг (длинная позиция) — покупать акции с целью их последующей продажи по более высокой цене.
2. Торговать только **15 лучших акций по объему на Московской бирже**.
3. Использовать два разных таймфрейма для тестов — это временные интервалы на которых отображается движение цен на графике финансового инструмента. Планирую использовать 5 минут и час. Это из-за того что **моё АПИ медленное**.

Откройте счёт в ВТБ Мои Инвестиции



## 2. Долгосрочное подтверждение: цена закрытия на часовом интервале выше часовой скользящей средней.



Требую выполнения обоих этих условий, гарантирую что акция будет иметь бычий импульс как на коротких, так и на длинных таймфреймах перед входом в позицию. Такое выравнивание двух таймфреймов помогает избегать покупок во время временного шума или незначительных колебаний на более коротком таймфрейме, отфильтровывая менее стабильные движения.

Условие продажи: трейлинг стоп, который предназначен для защиты прибыли и ограничения риска падения. Как работает лучше всего показано на картинке:

Откройте счёт в ВТБ Мои Инвестиции





## Бэктестинг моей торговой стратегии с помощью библиотеки backtrader на Python

Моя, описанная выше стратегия для двух таймфреймов на нескольких бумагах, выглядит в библиотеке backtrader на Python следующим образом:

strategy0\_ma\_5min\_hourly.py:

```
<code>import sys
sys.stdout.reconfigure(encoding='utf-8')

import backtrader as bt

# Стратегия скользящие средние на двух разных временных интервалах
class MovingAveragesOnDifferentTimeIntervalsStrategy(bt.Strategy):
    params = (
        ('ma_period_5min', 30),    # Период для скользящей средней на 5-минутках
        ('ma_period_hourly', 45),  # Период для скользящей средней на часовом инт
        ('trailing_stop', 0.03)    # Процент для трейлинг-стопа
    )

    def __init__(self):
```

```
        print(f'\nРасчет для параметров: {self.params.ma_period_5min} / {self.params.ma_period_hourly}
```

Откройте счёт в ВТБ Мои Инвестиции

```
# Для каждого инструмента добавляем скользящие средние по разным интерва
for i, data in enumerate(self.datas):
    if i % 2 == 0: # Четные индексы - 5-минутные данные
        ticker = data._name.replace('_5min', '')
        self.ma_5min[ticker] = bt.indicators.SimpleMovingAverage(data.c1
    else: # Нечетные индексы - часовые данные
        ticker = data._name.replace('_hourly', '')
        self.ma_hourly[ticker] = bt.indicators.SimpleMovingAverage(data.

# Переменные для отслеживания максимальной цены после покупки по каждому
self.buy_price = {}
self.max_price = {}
self.order = {} # Словарь для отслеживания ордеров по каждому инструмен

def next(self):
    # Для каждого инструмента проверяем условия покупки и продажи
    for i in range(0, len(self.datas), 2): # Проходим по 5-минутным данным
        ticker = self.datas[i]._name.replace('_5min', '')
        data_5min = self.datas[i]
        data_hourly = self.datas[i + 1]

        # Проверяем, есть ли открытый ордер для этого инструмента
        if ticker in self.order and self.order[ticker]:
            continue # Пропускаем, если есть открытый ордер

        # Проверяем условия покупки:
        # цена на 5 мин таймфрейме выше скользящей средней на 5 мин + часова
        if not self.getposition(data_5min): # Открываем сделку только если
            if data_5min.close[0] > self.ma_5min[ticker][0] and data_hourly.
                self.order[ticker] = self.buy(data=data_5min)
                self.buy_price[ticker] = data_5min.close[0]
                self.max_price[ticker] = self.buy_price[ticker]

        # Получаем текущий тикер и дату покупки
        buy_date = data_5min.datetime.date(0)
```

Откройте счёт в ВТБ Мои Инвестиции

```

current_price = data_5min.close[0]

# Обновляем максимальную цену, если текущая выше
if current_price > self.max_price[ticker]:
    self.max_price[ticker] = current_price

# Рассчитываем уровень стоп-лосса
stop_loss_level = self.max_price[ticker] * (1 - self.params.trai

# Проверяем условие для продажи по трейлинг-стопу
if current_price < stop_loss_level:
    self.order[ticker] = self.sell(data=data_5min)
    sell_date = data_5min.datetime.date(0)
    sell_time = data_5min.datetime.time(0)
    print(f"{sell_date} в {sell_time}: продажа за {current_price}

# Обрабатываем уведомления по ордерам
def notify_order(self, order):
    ticker = order.data._name.replace('_5min', '')

    if order.status in [order.Completed, order.Canceled, order.Margin]:
        self.order[ticker] = None # Очищаем ордер после завершения

```

Сделал переключатель одиночный тест или оптимизация: singleTest / optimization для  
основного файла запуска: SingleTestOrOptimization = "optimization"

Основной файл запуска main.py:

```

import sys
import time
sys.stdout.reconfigure(encoding='utf-8')

from datetime import datetime
from src.data_loader import load_data_for_ticker, load_ticker_mapping

import pandas as pd
import backtrader as bt

```

Откройте счёт в ВТБ Мои Инвестиции

```
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 1000)

# Начало времени
start_time = time.perf_counter()

# Путь к JSON файлу с сопоставлениями
mapping_file = "./data/+mappings.json"

# Загрузка сопоставлений тикеров
ticker_mapping = load_ticker_mapping(mapping_file)

# Промежуточное время выполнения
total_end_time = time.perf_counter()
elapsed_time = total_end_time - start_time
print(f"Промежуточное время выполнения: {elapsed_time:.4f} секунд.")

current_time = datetime.now().strftime("%Y-%m-%d %H-%M") # Генерируем текущее вр

# Следующая часть кода запускается только если это основной модуль
if __name__ == '__main__': # Исправление для работы с multiprocessing

    # Создаем объект Cerebro
    cerebro = bt.Cerebro(optreturn=False)

    # Получаем количество бумаг в ticker_mapping.items()
    num_securities = len(ticker_mapping.items())

    # Рассчитываем процент капитала на одну бумагу
    percent_per_security = 100 / num_securities
    print(f"Процент капитала на одну бумагу: {percent_per_security:.2f}%")

    # Условия капитала
    cerebro.broker.set_cash(100000) # Устанавливаем стартовый капитал
    cerebro.broker.setcommission(commission=0.005) # Комиссия 0.5%
```

Откройте счёт в ВТБ Мои Инвестиции

```
# Загрузка данных с таймфреймами 5 минут и час
data_5min, data_hourly = load_data_for_ticker(ticker)

# Пропуск, если данные не были загружены
if data_5min is None or data_hourly is None:
    continue

# Добавляем 5-минутные данные в Cerebro
data_5min_bt = bt.feeds.PandasData(dataname=data_5min, timeframe=bt.Time
cerebro.adddata(data_5min_bt, name=f"{ticker}_5min")

# Добавляем часовые данные в Cerebro
data_hourly_bt = bt.feeds.PandasData(dataname=data_hourly, timeframe=bt.

# Совмещаем графики 5 минут и часа на одном виде
data_hourly_bt.plotinfo.plotmaster = data_5min_bt # Связываем графики
data_hourly_bt.plotinfo.sameaxis = True           # Отображаем на той ж
cerebro.adddata(data_hourly_bt, name=f"{ticker}_hourly")

# Переключатель одиночный тест или оптимизация
SingleTestOrOptimization = "optimization" # singleTest / optimization

if SingleTestOrOptimization == "singleTest":
    print(f"{current_time} Проводим одиночный тест стратегии.")

# Добавляем стратегию для одичного теста MovingAveragesOnDifferentTimeIn
cerebro.addstrategy(MovingAveragesOnDifferentTimeIntervalsStrategy,
                    ma_period_5min = 30,      # Период для скользящей средней на
                    ma_period_hourly = 45,    # Период для скользящей средней на
                    trailing_stop = 0.03)     # Процент для трейлинг-стопа

# Writer только для одиночного теста для вывода результатов в CSV-файл
cerebro.addwriter(bt.WriterFile, csv=True, out=f"./results/{current time
```

Откройте счёт в ВТБ Мои Инвестиции

```
cerebro.addanalyzer(btanalyzers.SharpeRatio, _name='sharpe_ratio', timef
cerebro.addanalyzer(btanalyzers.SQN, _name='sqn')
cerebro.addanalyzer(btanalyzers.PyFolio, _name='PyFolio')
```

```
# Запуск тестирования
```

```
results = cerebro.run(maxcpus=1) # Ограничение одним ядром для избежани
```

```
# Выводим результаты анализа одиночного теста
```

```
print(f"\nОкончательная стоимость портфеля: {cerebro.broker.getvalue()}"
```

```
returnsAnalyzer = results[0].analyzers.returns.get_analysis()
```

```
print(f"Годовая/нормализованная доходность: {returnsAnalyzer['rnorm100']
```

```
drawdownAnalyzer = results[0].analyzers.drawdown.get_analysis()
```

```
print(f"Максимальное значение просадки: {drawdownAnalyzer['max']['drawdo
```

```
trade_analyzer = results[0].analyzers.trade_analyzer.get_analysis()
```

```
print(f"Всего сделок: {trade_analyzer.total.closed} шт.")
```

```
print(f"Выигрышные сделки: {trade_analyzer.won.total} шт.")
```

```
print(f"Убыточные сделки: {trade_analyzer.lost.total} шт.")
```

```
sharpe_ratio = results[0].analyzers.sharpe_ratio.get_analysis().get('sha
```

```
print(f"Коэффициент Шарпа: {sharpe_ratio}")
```

```
sqnAnalyzer = results[0].analyzers.sqn.get_analysis().get('sqn')
```

```
print(f"Мера доходности с поправкой на риск: {sqnAnalyzer}")
```

```
# Время выполнения
```

```
total_end_time = time.perf_counter()
```

```
elapsed_time = (total_end_time - start_time) / 60
```

```
print(f"\nВремя выполнения: {elapsed_time:.4f} минут.")
```

```
# Построение графика для одиночного теста
```

```
cerebro.plot()
```

```
else:
```

```
print(f"{current_time} Проводим оптимизацию стратегии.")
```

```
# Оптимизация стратегии start date = 2024-10 MovingAveragesOnDifferentTi
```

Откройте счёт в ВТБ Мои Инвестиции

```

# Добавляем анализаторы
cerebro.addanalyzer(btanalyzers.TradeAnalyzer, _name='trade_analyzer')
cerebro.addanalyzer(btanalyzers.DrawDown, _name="drawdown")
cerebro.addanalyzer(btanalyzers>Returns, _name="returns")
cerebro.addanalyzer(btanalyzers.SharpeRatio, _name='sharpe_ratio', timeframe='1m')
cerebro.addanalyzer(btanalyzers.SQN, _name='sqn')

# Запуск тестирования
results = cerebro.run(maxcpus=1) # Ограничение одним ядром для избежани

# Выводим результаты оптимизации
par_list = [ [
    # MovingAveragesOnDifferentTimeIntervalsStrategy
    x[0].params.ma_period_5min,
    x[0].params.ma_period_hourly,
    x[0].params.trailing_stop,

    x[0].analyzers.trade_analyzer.get_analysis().pnl.net.tot
    x[0].analyzers.returns.get_analysis()['rnorm100'],
    x[0].analyzers.drawdown.get_analysis()['max']['drawdown']
    x[0].analyzers.trade_analyzer.get_analysis().total.close
    x[0].analyzers.trade_analyzer.get_analysis().won.total,
    x[0].analyzers.trade_analyzer.get_analysis().lost.total,
    x[0].analyzers.sharpe_ratio.get_analysis()['sharperatio']
    x[0].analyzers.sqn.get_analysis().get('sqn')
] for x in results]

# MovingAveragesOnDifferentTimeIntervalsStrategy
par_df = pd.DataFrame(par_list, columns = ['ma_period_5min', 'ma_period_

# Формируем имя файла с текущей датой и временем
filename = f"./results/{current_time}_optimization.csv"
# Сохраняем DataFrame в CSV файл с динамическим именем
par_df.to_csv(filename, index=False)

```

Откройте счёт в ВТБ Мои Инвестиции



```
print(f"\nВремя выполнения: {elapsed_time:.4f} минут.")
```

```
# Общее время выполнения
```

```
total_end_time = time.perf_counter()
```

```
elapsed_time = (total_end_time - start_time) / 60
```

```
print(f"\nОбщее время выполнения: {elapsed_time:.4f} минут.")</code>
```

В данные загрузил котировки за октябрь 2024:

1. AFLT\_1hour.csv
2. AFLT\_5min.csv
3. EUTR\_1hour.csv
4. EUTR\_5min.csv
5. GAZP\_1hour.csv
6. GAZP\_5min.csv
7. MTLR\_1hour.csv
8. MTLR\_5min.csv
9. RNFT\_1hour.csv
10. RNFT\_5min.csv
11. ROSN\_1hour.csv
12. ROSN\_5min.csv
13. RUAL\_1hour.csv
14. RUAL\_5min.csv
15. SBER\_1hour.csv
16. SBER\_5min.csv
17. SGZH\_1hour.csv
18. SGZH\_5min.csv
19. SNGSP\_1hour.csv
20. SNGSP\_5min.csv
21. UWGN\_1hour.csv
22. UWGN\_5min.csv
23. VKCO\_1hour.csv
24. VKCO\_5min.csv
25. VTBR\_1hour.csv
26. VTBR\_5min.csv

Время выполнения оптимизации для таких параметров составило 74 минуты!

Откройте счёт в ВТБ Мои Инвестиции

```
ma_period_hourly=range(15, 61, 2), # Диапазон для часовой скол
trailing_stop=[0.03]) # Разные проценты для трейли
```

```

124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	ACROS
4	10	23	0.03	-5820.012958 -0.000984 0.253991 100 24 84 -2.380005 -4.102218
...	...	...	...	...
248	60	51	0.03	-4795.729618 -0.000818 5.717055 86 21 65 -2.616581 -3.731260
249	60	53	0.03	-6430.731804 -0.000758 5.581071 84 21 63 -2.455662 -3.255305
250	60	55	0.03	-4374.374785 -0.000748 5.512912 83 21 62 -2.582423 -3.194545
251	60	57	0.03	-4337.500821 -0.000741 5.471727 80 20 60 -2.586523 -3.250978
252	60	59	0.03	-4846.760571 -0.000692 5.346015 79 20 59 -2.580299 -3.029657

[253 rows x 11 columns]

Время выполнения: 74.5854 минут.

Общее время выполнения: 74.5854 минут.

Для того чтобы визуально представить результаты оптимизации написал модуль, который строит трехмерный график.

Модуль 3dchart.py:

```

import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from matplotlib.widgets import Slider
from datetime import datetime

```

```
# Чтение данных из CSV файла
```

```
data = pd.read_csv('./results/2024-11-12 16-12_optimization_2024-10_MovingAverag
```

```
parameter1 = 'ma_period_5min'
```

```
parameter2 = 'ma_period_hourly'
```

```
# Извлечение необходимых колонок для построения графика
```

Откройте счёт в ВТБ Мои Инвестиции

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Построение поверхности с использованием триангуляции
surf = ax.plot_trisurf(x, y, z, cmap='viridis', edgecolor='none')

# Подписи к осям
ax.set_xlabel(parameter1)
ax.set_ylabel(parameter2)
ax.set_zlabel('PNL Net')

# Заголовок графика
current_time = datetime.now().strftime("%Y-%m-%d %H:%M") # Генерируем текущее вр
ax.set_title(f"3D Optimization Chart, {current_time}")

# Добавление плоскости, которая будет двигаться вдоль оси Z
# Начальное значение плоскости по оси Z
z_plane = np.mean(z)

# Плоскость - запоминаем ее как отдельный объект
x_plane = np.array([[min(x), max(x)], [min(x), max(x)]])
y_plane = np.array([[min(y), min(y)], [max(y), max(y)]])
z_plane_values = np.array([[z_plane, z_plane], [z_plane, z_plane]])

# Отображение плоскости
plane = ax.plot_surface(x_plane, y_plane, z_plane_values, color='red', alpha=0.5)

# Создание слайдера для управления позицией плоскости по оси Z
ax_slider = plt.axes([0.25, 0.02, 0.50, 0.03], facecolor='lightgoldenrodyellow')
z_slider = Slider(ax_slider, 'Z Plane', min(z), max(z), valinit=z_plane)

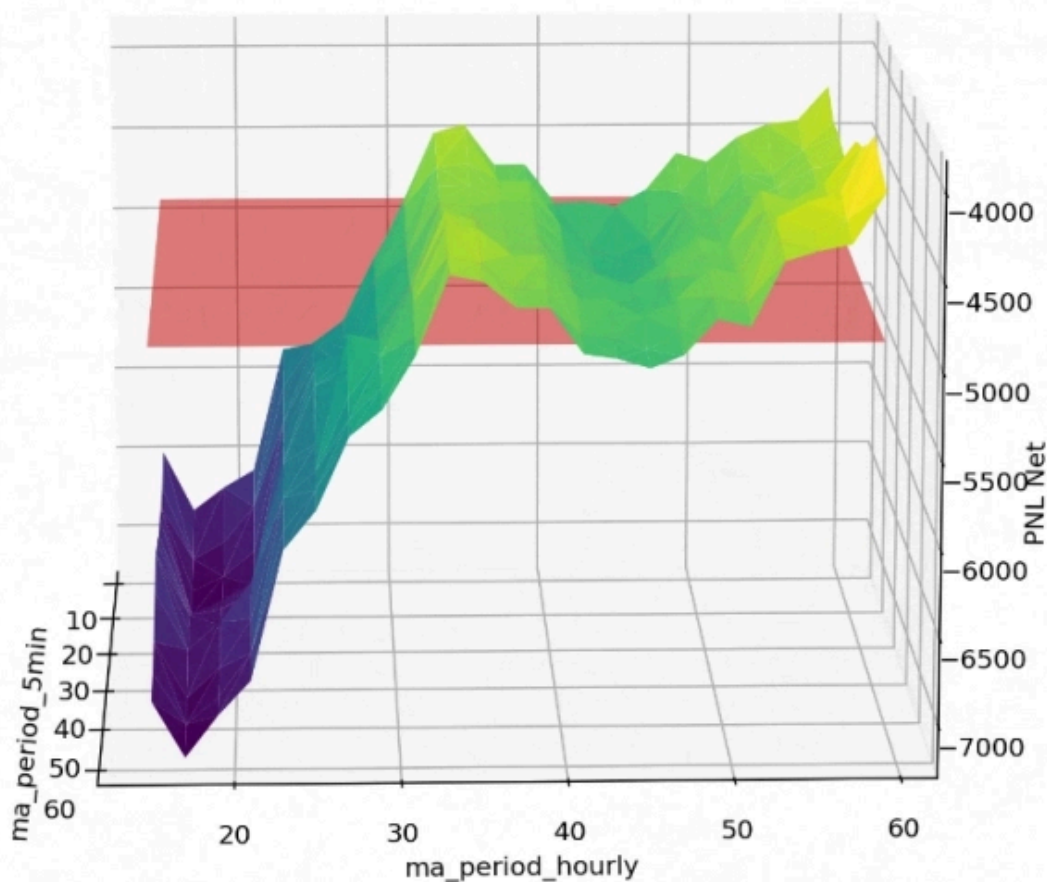
# Функция обновления положения плоскости при перемещении слайдера
def update(val):
    new_z_plane = z_slider.val
    z_plane_values[:, :] = new_z_plane # Обновляем значения Z для плоскости
```

Откройте счёт в ВТБ Мои Инвестиции

```
z_slider.on_changed(update)
```

```
# Отображение графика  
plt.show()</code>
```

Результат оптимизации в виде графика:



[Гифка здесь](#). Сюда не смог вставить.

## Выводы из этой оптимизации

Цифры по шкале Z показывают лишь степень убытков в рублях. Они со знаком минус.

Вы можете сами полностью повторить мой опыт потому что код загружен на GitHub:[https://github.com/empenoso/SilverFir-TradingBot\\_backtesting](https://github.com/empenoso/SilverFir-TradingBot_backtesting)

Откройте счёт в ВТБ Мои Инвестиции

2. Курвефитинг, подгонка под историю. Не хочу вводить много параметров, чтобы этого избежать. Переобучение прошлыми данными: если стратегия хорошо работает на исторических данных, но плохо на будущих данных в режиме скользящего окна, она может быть слишком адаптирована к историческим моделям, которые не будут повторяться.
3. Транзакционные затраты: хорошо, если тестирование учитывает реалистичное проскальзывание, комиссии и спреды.

## Будущие шаги — где искать прибыльные торговые стратегии

Я хочу использовать подход скользящего окна — когда данные разбиваются на более мелкие последовательные периоды например по месяцам, за которым следует период тестирования вне этой выборки. Например, оптимизация идёт на месячных данных, а тестировать уже на следующем месяце. То есть происходит сдвиг вперед: после каждого периода тестирования окно «скользит» вперед на указанный интервал, и процесс повторяется. Таким образом, каждый сегмент данных используется как для обучения, так и для тестирования с течением времени, но никогда одновременно. Это помогает проверить, что стратегия работает стабильно в меняющихся рыночных условиях.

Также планирую использовать Technical Analysis of STOCKS & COMMODITIES для поиска новых идей. Их советы трейдерам [доступны в открытом доступе](#).

А ещё планирую использовать ChatGPT, отправляя запросы вроде:

*Действуй как опытный издатель. Отобрази 10 ведущих авторов в области алгоритмической торговли на рынке Америки. Для каждого автора перечисли три самые популярные книги, включая сведения о книге (дату публикации, издателя и ISBN), и предоставь русские переводы для каждого названия книги.*

Откройте счёт в ВТБ Мои Инвестиции

- "Machine Learning for Asset Managers"
    - (2020, Cambridge University Press, ISBN: 978-1108492453)
    - Russian title: "Машинное обучение для управляющих активами"
  - "Quantitative Research in Trading"
    - (2013, No Starch Press, ISBN: 978-1118457573)
    - Russian title: "Количественные исследования в торговле"
- ### 3. Yves Hilpisch
- "Python for Finance: Mastering Data-Driven Finance"
    - (2018, O'Reilly, ISBN: 978-1492024330)
    - Russian title: "Python для финансов: мастерство в финансах, основанных на данных"
  - "Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration, and Hedging"
    - (2015, Wiley, ISBN: 978-1119037996) ↓
    - Russian title: "Аналитика деривативов с Python: анализ данных, модели, симуляция,



Сообщить ChatGPT



ChatGPT может допускать ошибки. Рекомендуем проверять важную информацию.

и дальше после ответа:

*Действуй как опытный пользователь библиотеки backtrader на Python. Хочу использовать торговую стратегию из книги Yves Hilpisch «Python for Finance: Mastering Data-Driven Finance» для тестов.*

Добавляй все комментарии на русском языке, продолжай со мной общение на английском.

Откройте счёт в ВТБ Мои Инвестиции

English.



Память обновлена

Certainly! I'll provide you with a basic Backtrader implementation for the strategy described in Hilpisch's *Python for Finance*. Typically, the strategy in the book uses a mean-reversion approach with Bollinger Bands and signals based on crossings of price with upper and lower bands.

Here's an example setup in Backtrader. Each line includes Russian comments as requested:

python

Копировать код

```
import backtrader as bt

# Создаем класс стратегии, наследуя его от класса bt.Strategy
class MeanReversionStrategy(bt.Strategy):
    params = (
        ("period", 20),          # Период для расчета индикаторов
        ("devfactor", 2.0),      # Коэффициент для отклонения в полосах Боллинджера
    )
```



Сообщить ChatGPT



ChatGPT может допускать ошибки. Рекомендуем проверять важную информацию.

И дальше подобные промты.

## Итоги

Несмотря на то, что первоначальный выбор стратегии на двух разных таймфреймах и сразу для 15 активов был не самый удачный — впереди ещё очень большое поле исследований и тестов.

Автор: [Михаил Шардин](#)

18 ноября 2024 г.

трейдинг

торговые роботы

4.7K


13

30

21

Откройте счёт в ВТБ Мои Инвестиции





257

2 502

с 23 января 2019

+HreHDn1F5CZjN...

+ Подписаться

## 30 КОММЕНТАРИЕВ

Сначала старые ▾



Ramha

18 ноября 2024, 06:00

сочинять и тестировать модели удобно в Амиброкер [www.amibroker.com/](http://www.amibroker.com/)

— Показать 1 ответ



SergeyJu

18 ноября 2024, 09:04

Зачем Вами все эти костыли. Вы умеете программировать, ну и пишете все сами на удобном для Вас языке. Питоне, Си, без разницы. Да, и базу данных надо прикрутить желательно тоже стандартную.

— Показать 4 ответа



Просто трейдер

18 ноября 2024, 11:28

Какая каша из всего. У тебя почти всё из списка не работает с Московской. На Москве это МетаТрейдер, СтокШарп, ТС Лаб и старина Квик.

— Показать 6 ответов



BorisN

18 ноября 2024, 14:08

То есть, на реальных деньгах ещё не проверяли?

— Показать 2 ответа



Откройте счёт в ВТБ Мои Инвестиции

Напишите комментарий...



ОТПРАВИТЬ

## Читайте на SMART-LAB

ЗА ПОЛУГОДИЕ, ЗАКОНЧИВШЕЕСЯ 30 ИЮНЯ 2025 ГОДА (НЕАУДИРОВАННЫЙ)

(в миллионах российских рублей, за исключением прибыли на акции)

	Примечание	30 июня 2025 года	30 июня 2024 года
ВЫРУЧКА	16	171 233	201 884
СЕБЕСТОИМОСТЬ РЕАЛИЗАЦИИ	17	(164 266)	(160 252)
<b>ВАЛОВАЯ ПРИБЫЛЬ</b>		<b>26 969</b>	<b>41 632</b>
Коммерческие, общезаведомые и административные расходы	18	(15 722)	(18 177)
Почтенные операционные доходы	19	3 385	185
Почтенные операционные расходы		(137)	(115)
Изменение справедливой стоимости инвестиционной недвижимости		125	-
<b>ОПЕРАЦИОННЫЙ УБЫТОК (ПРИБЫЛЬ)</b>		<b>(7 539)</b>	<b>3 720</b>
Финансовые доходы	20	505	443
Финансовые расходы	20	(16 432)	(16 572)
<b>УБЫТОК ДО НАЛОГА НА ПРИБЫЛЬ</b>		<b>(13 466)</b>	<b>(12 394)</b>
Доходы по налогу на прибыль		8 527	2 036
Итого чистый убыток и чистый финансовый результат на период		<b>(4 939)</b>	<b>(10 358)</b>

## М.Видео - есть ли в мультивселенной компания, где миноритариям что-то достанется?

М.Видео меняет формат допэмиссии. Ранее акционеры одобрили допэмиссию до 500 млн новых акций по закрытой подписке — о...



Mozgovik

03:51

## ДАЙДЖЕСТ ПО РЕЙТИНГОВЫМ ДЕЙСТВИЯМ В ВЫСОКОДОХОДНОМ СЕКТОРЕ, ПОРТФЕЛЕ PROBONDS ВДО И РОЗНИЧНЫХ ИНВЕСТИЦИОННЫХ ОБЛИГАЦИЙ ЗА...

● ПАО «ЕВРОТРАНС» АКРА подтвердило кредитный рейтинг на уровне A-(RU) ПАО «ЕвроТранс» — один из крупнейших независимых топливных операторов на рынке Московского региона....



Иволга Капитал

13.09.2025

## Банк России снизил ключевую ставку до 17% — что делать? Мнение аналитиков

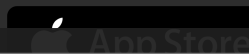
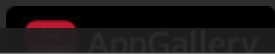
Сегодня на заседании Банк России снизил ключевую ставку на 1 п.п., до 17%. Теоретически, для рынков акций и облигаций — это определенно благоприятное событие, но на практике рынок...



БКС Мир инвестиций

13.09.2025

Установите приложение Смартлаба:



Откройте счёт в ВТБ Мои Инвестиции

[О smart-lab](#) [Реклама](#) [Полная версия](#)



Московская Биржа является спонсором ресурса smart-lab.ru  
Источник: [ПАО Московская Биржа](#)

Откройте счёт в ВТБ Мои Инвестиции