

Streaming Financial Data and Plots with Plotly

Yves Hilpisch

Plotcon, Oakland, 04. May 2017



SERVICES

for financial institutions globally



EVENTS

for Python quants & algorithmic traders



TRAINING

about Python for finance
& algorithmic trading



CERTIFICATION

in cooperation with university



BOOKS

about Python and
finance



PLATFORM

for browser-based
data analytics

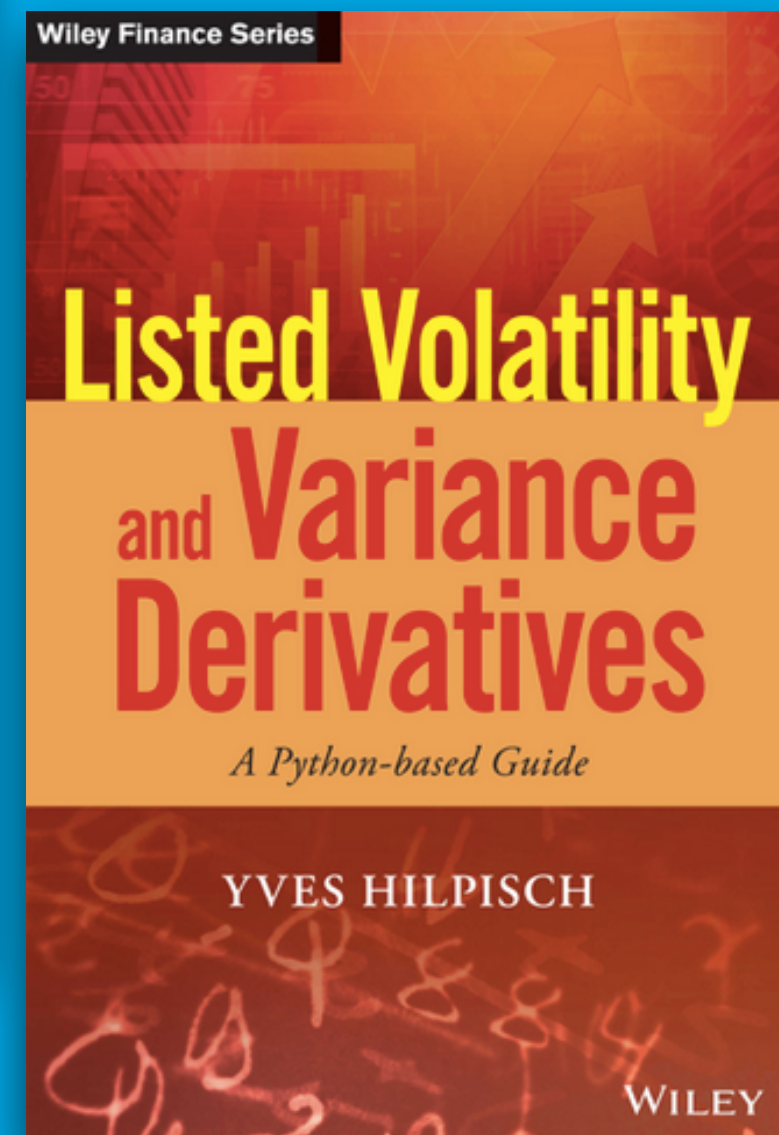
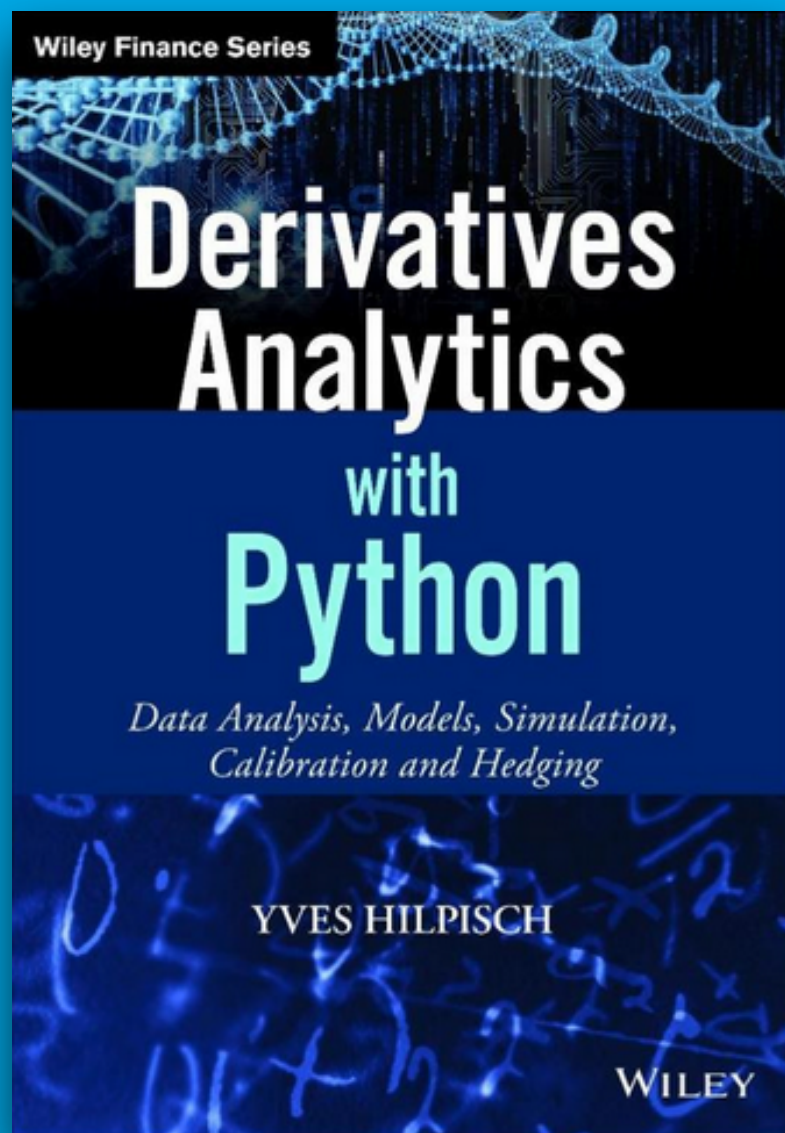


OPEN SOURCE

Python library
for financial analytics







Quant Platform

<https://pyalgo.pqp.io/nb/portal/login>

Default Kernel

Python 2.7

Python 3.4

R

Logout

Python for Algorithmic Trading

9. Stock Trading with Interactive Brokers

9.1. Introduction

9.2. Setting up an Account

9.3. Python and the IB API

9.4. A Wrapper Class for the IB API

9.5. Retrieving Historical Data from IB

9.6. Working with Streaming Data from IB

9.7. Retrieving Account Information

9.8. Implementing Trading Strategies in Real-Time

9.9. Conclusions

9.10. Further Resources

9.11. Python Scripts

10. Algorithmic Trading of Cryptocurrencies

10.1. Introduction

10.2. Cryptocurrency Exchanges

10.3. RESTful APIs and Streaming APIs

10.4. Trading Strategies for Cryptocurrencies

10.5. Implementing Trading Strategies in Real-Time

10.6. Conclusions

10.7. Further Resources

10.8. Python Scripts

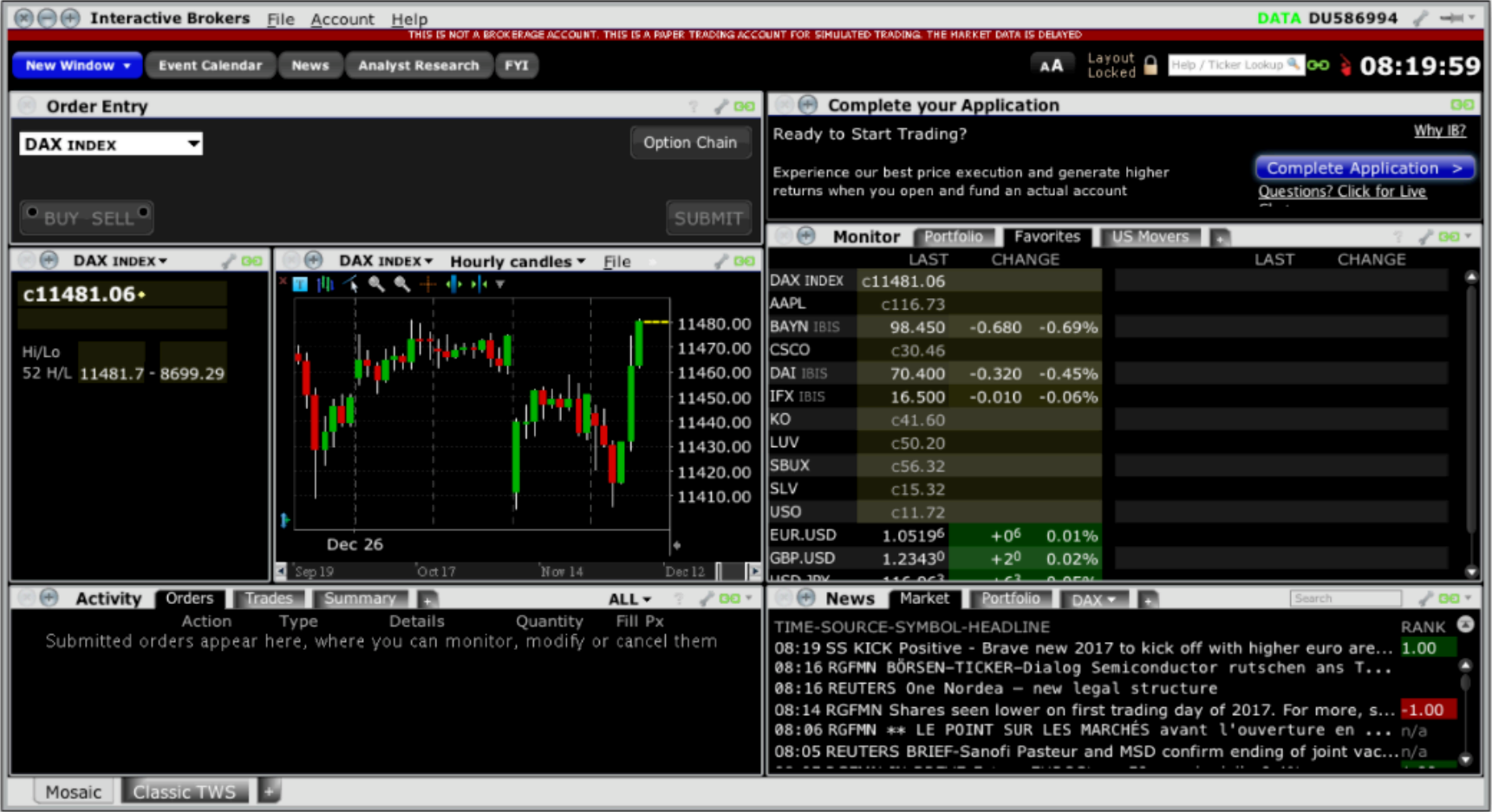
11. Automating Trading Operations

11.1. Introduction

11.2. Capital Management Strategies

11.3. Risk Management

Once logged in, you can then download the TWS application for your operating system. Starting the application then requires the previously chosen user name and password. TWS then might show up as in [Trader Workstation after login with trial credentials](#) on your desktop.



The screenshot shows the Interactive Brokers Trader Workstation (TWS) interface. At the top, there's a menu bar with 'File', 'Account', and 'Help'. Below it, a status bar indicates 'THIS IS NOT A BROKERAGE ACCOUNT. THIS IS A PAPER TRADING ACCOUNT FOR SIMULATED TRADING. THE MARKET DATA IS DELAYED.' The main interface is divided into several panels. On the left, there's an 'Order Entry' panel for the 'DAX INDEX' with 'BUY' and 'SELL' buttons. In the center, there's a 'DAX INDEX' window showing a candlestick chart for 'Hourly candles' on 'Dec 26'. On the right, there's a 'Complete your Application' window with a 'Complete Application' button. Below that, there's a 'Monitor' window showing a list of stocks with their 'LAST' and 'CHANGE' values. At the bottom, there's an 'Activity' window showing a table of submitted orders.

Figure 58. Trader Workstation after login with trial credentials

The arrangement of the different panels of TWS might be changed or new windows might pop up depending on what you request from the application. [TWS break out window with option chain data](#) shows a break out window with option chain

PROGRAM DIRECTOR

Dr. Yves J. Hilpisch is founder and managing partner of The Python Quants (<http://tpq.io>), a group focusing on the use of open source technologies for financial data science, algorithmic trading and computational finance. He is the author of the books:

- Python for Finance (O'Reilly)
- Python for Analytics with Python (Wiley)
- Listed Volatility and Variance Derivatives (Wiley)

He has written the financial analytics library DX Analytics (<http://dx-analytics.com>) and organizes conferences and Meetup events about Python for finance and algorithmic trading in Frankfurt, London and New York. He has given keynote speeches at technology conferences in the United States, Europe and Asia.

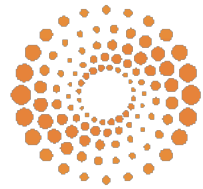


UNIVERSITY CERTIFICATE IN PYTHON FOR ALGORITHMIC TRADING



The Python Quants GmbH
66333 Voelklingen
Germany
T/F +49 3212 112 91 94
<http://training.tpq.io>
training@tpq.io

April 2017



THOMSON REUTERS

FitchLearning

CQF | INSTITUTE

htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

**historical, unstructured data
(news, tweets)**

**historical, structured data
(stock prices, P&L)**

**real-time, unstructured data
(news, tweets)**

**real-time, structured data
(stock ticks, exchange rates)**

**real-time processing,
real-time visualization**


```
#
# Simple Tick Data Server with
# ZeroMQ
#
import zmq
import time
import random

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind('tcp://0.0.0.0:5555')

AAPL = 100.

while True:
    AAPL += random.gauss(0, 1) * 0.5
    msg = 'AAPL %s' % AAPL
    tick_server.py [+
```

```
#
# Simple Tick Data Client with
# ZeroMQ
#
import zmq
import datetime

context = zmq.Context()
socket = context.socket(zmq.SUB)
socket.connect('tcp://0.0.0.0:5555')
socket.setsockopt_string(zmq.SUBSCRIBE, 'AAPL')

while True:
    msg = socket.recv_string()
    t = datetime.datetime.now()
    print('%s | %s' % (t, msg))

tick_client.py
```

× IPython: live/data (python3.6)

AAPL	107.15636235397254
AAPL	107.18612019583905
AAPL	107.4983187955743
AAPL	107.2640892475144
AAPL	107.68358829560407
AAPL	106.9232056802307
AAPL	106.55017297488794
AAPL	105.97708319698597
AAPL	106.00856053822193
AAPL	105.37221723045396
AAPL	105.09251644774177
AAPL	104.9267694947986
AAPL	105.03306681222703
AAPL	105.1223727550806
AAPL	105.29880694705703
AAPL	105.438670667864
AAPL	105.60426198517378

1

```

X root@pythonquants02: ~ (python3.6)

```

Time	Price	Volume	Symbol	Price	Volume	Symbol
2017-05-01 23:51:44.010545			AAPL	106.94730057503057		
2017-05-01 23:51:44.184665			AAPL	107.15636235397254		
2017-05-01 23:51:44.663153			AAPL	107.18612019583905		
2017-05-01 23:51:44.707051			AAPL	107.4983187955743		
2017-05-01 23:51:45.066229			AAPL	107.2640892475144		
2017-05-01 23:51:45.433200			AAPL	107.68358829560407		
2017-05-01 23:51:46.315111			AAPL	106.9232056802307		
2017-05-01 23:51:47.040770			AAPL	106.55017297488794		

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

2017-05-01 23

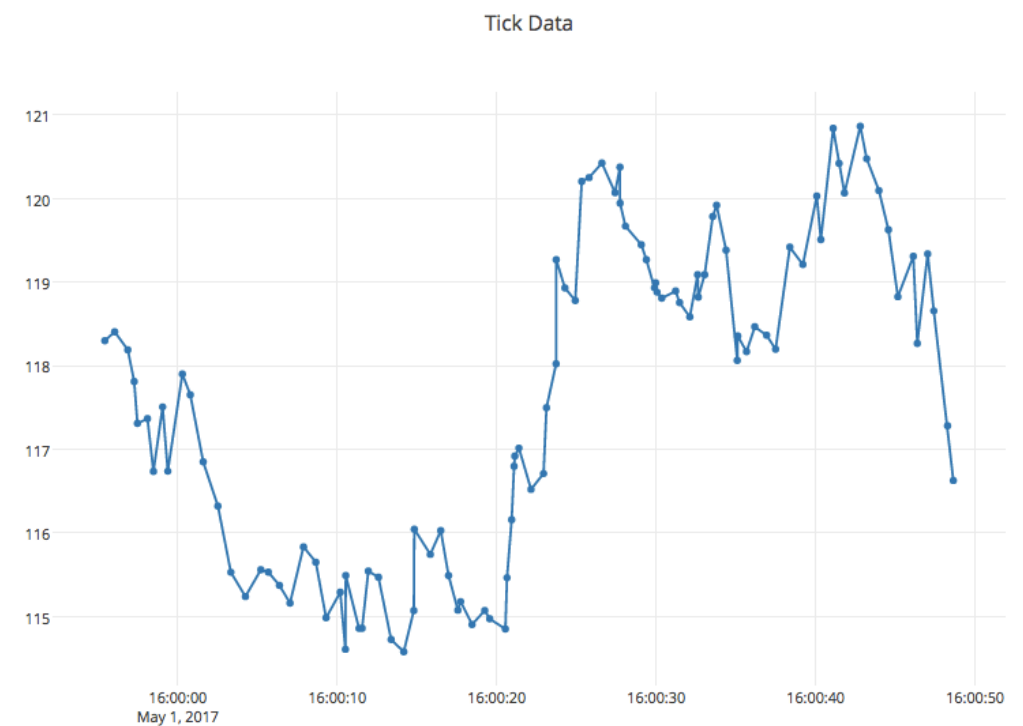
121

120

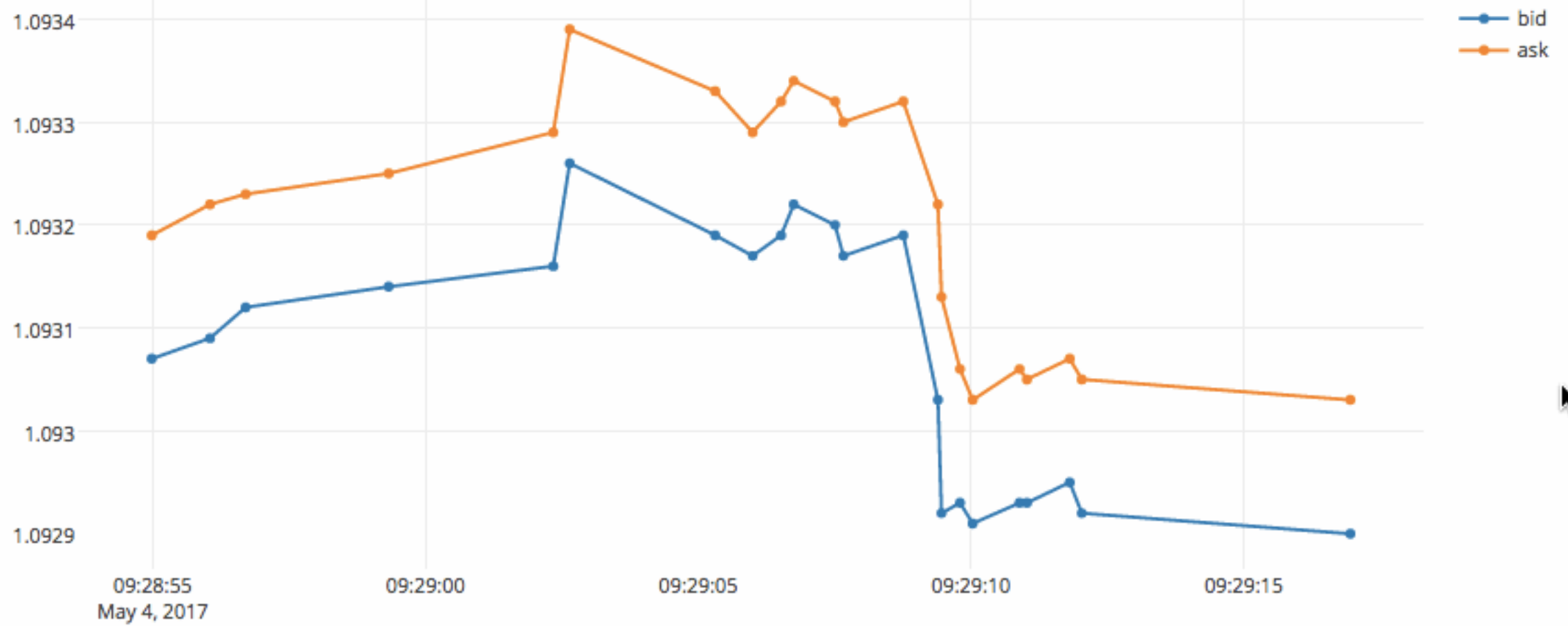
119

Tick Data

1



EUR_USD



EDIT CHART

The Python Quants GmbH

Dr. Yves J. Hilpisch

+49 3212 112 9194

<http://tpq.io> | team@tpq.io

@dyjh

