Twitter Thread by PyQuant News

PyQuant News
@pyquantnews

In 2012, my first options trade lost $9,000.

12 months later I was making $1,100 per week trading in my free time.

What changed?

I read 20 books on options and finished a master's degree.

But what took my game to the next level was Python.

Here's the code I still use today:

A quick primer on options in case you're not familiar:

• Over $450 billion in notional trades DAILY
• 39 million options contracts trade DAILY
• 25% of total options trading is from retail

So how do we make money?

What Are Options?

Options are standardized derivatives contracts that convey the buyer the right (but not obligation) to buy the underlying security in the case of a call option or sell the underlying security in the case of a put option at a given price before a given date (for American style options) or on a given date (European style options).

There are many types of options of which only a few are available for retail traders. "Exotics" exist that are generally large dollar trades (millions of US$) and traded among investment banks. These are generally custom built by quants to help solve a customer's specific financial problem.

We will focus on plain vanilla, european style, equity options so we can use the famous Black-Scholes pricing formula. More on Black-Scholes a bit later.

Most derivatives have a payoff function which describes the value at the end of the life of the contract (expiration). For a call option, the payoff can be expressed simply as:

$$C(S, K) = max(S - K, 0)$$

While the payoff for a put option can be expressed as:

$$P(K, S) = max(K - S, 0)$$

Where S and K are defined above. In this case, K remains fixed during the life of a contract while the underlying stock price, S, fluctuates..

Across a range of potential S values, we can form what is commonly known as the payoff (or risk profile or PnL chart) for an option at expiration.

The challenge for quants is to figure out what the price of the option is *before* expiration. This is where option pricing models like Black-Scholes come into play.

@pyquantnews

Trade options with a simple, 3-part framework:

1. Design your risk
2. Value the position
3. Measure and monitor

Now pair this framework with Python and you get a potent combination for making money trading options.

Let's dig in:

To design our risk profile, we need the option payoff.

Let's start by defining the variables we need:

• Stock price
• Strike price
• Time to expiration
• Interest rate
• Volatility

We also define market prices for demonstrating trades.

```python
# underlying stock price
S = 45.0

# series of underlying stock prices to demonstrate a payoff profile
S_ = np.arange(35.0, 55.0, 0.01)

# strike price
K = 45.0

# time to expiration (you'll see this as T-t in the equation)
t = 164.0 / 365.0

# risk free rate (there's nuance to this which we'll describe later)
r = 0.02

# volatility (latent variable which is the topic of this talk)
vol = 0.25

# black scholes prices for demonstrating trades
atm_call_premium = 3.20
atm_put_premium = 2.79

otm_call_premium = 1.39
otm_put_premium = 0.92
```


Remember our payoff functions for calls and puts?

- call = max(S - K, 0)
- put = max(K - S, 0)

We can define them in Python in one line of code each.

And here's where it gets really interesting.

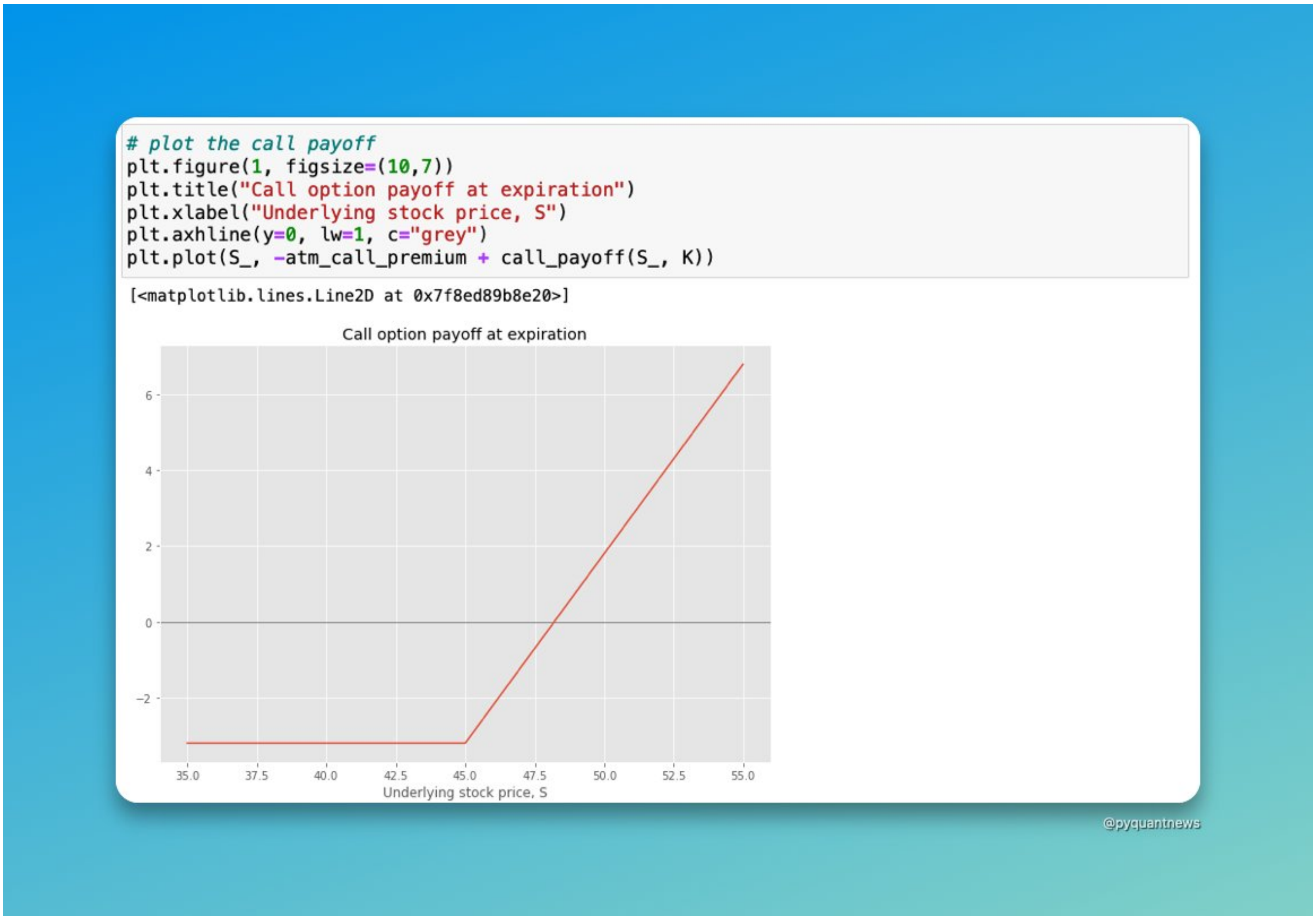
```python
# use a lambda for a payoff functions
# equivelant to:
#
# def call_payoff(S, K):
#     return np.maximum(S - K, 0.0)
call_payoff = lambda S, K: np.maximum(S_ - K, 0.0)

# equivelant to:
#
# def put_payoff(S, K):
#     return np.maximum(K - S, 0.0)
put_payoff = lambda S, K: np.maximum(K - S_, 0.0)
```

Using NumPy, we define a series of stock prices.

This gives us the payoff at each of these prices.

Then we plot the payoff against the stock prices and we have our famous hockey stick payoff chart for a call option!

We do the same thing for a put option.



We can clearly see that as the stock price decreases in price, the value of our put option increases.

Here's the amazing part of this...

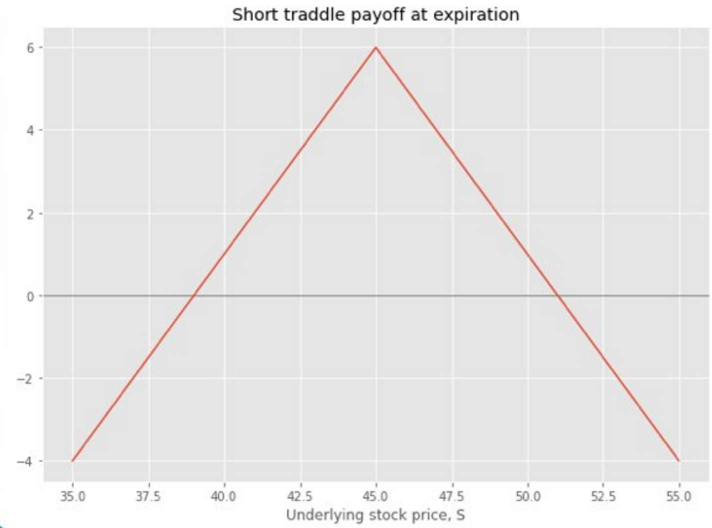
Using this framework, we can construct any complex options position we want.

For example, I modeled a short straddle with breakeven points at $38.75 and $51.25 and max profit when the stock price is at $45.

Butterfly spread?

```python
# plot a short straddle payoff
short_straddle = -call_payoff(S_, K) - put_payoff(S_, K)
short_straddle_premium = atm_call_premium + atm_put_premium
plt.figure(4, figsize=(10, 7))
plt.title("Short traddle payoff at expiration")
plt.xlabel("Underlying stock price, S")
plt.axhline(y=0, lw=1, c="grey")
plt.plot(S_, short_straddle_premium - long_straddle)
```

```
[<matplotlib.lines.Line2D at 0x7f8eb95f3220>]
```



@pyquantnews

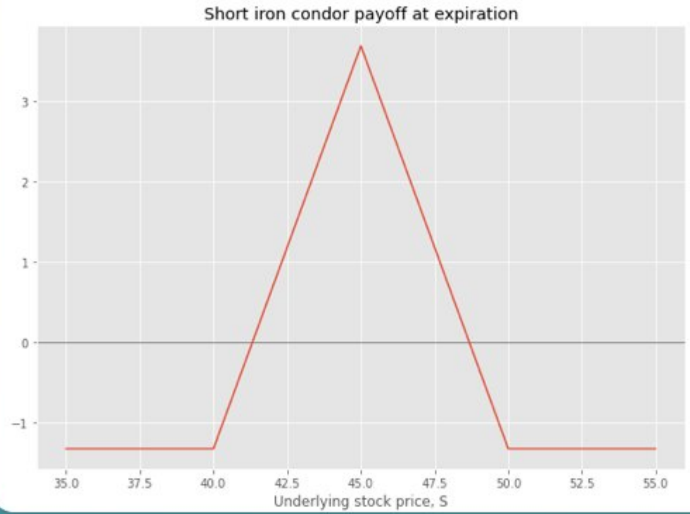
A butterfly spread is similar to a straddle except the loss is capped.

Because you're buying additional options to protect the downside, the premium collected is less than a straddle.

This is clear when you design your risk upfront.

```
# plot a short iron condor payoff
short_iron_condor = (
    call_payoff(S_, K + 5)
    - call_payoff(S_, K)
    - put_payoff(S_, K)
    + put_payoff(S_, K - 5)
)
short_iron_condor_premium = (
    -otm_call_premium + atm_call_premium + atm_put_premium - otm_put_premium
)
plt.figure(5, figsize=(10, 7))
plt.title("Short iron condor payoff at expiration")
plt.xlabel("Underlying stock price, S")
plt.axhline(y=0, lw=1, c="grey")
plt.plot(S_, short_iron_condor_premium + short_iron_condor)
```

```
[<matplotlib.lines.Line2D at 0x7f8ee9025eb0>]
```



The value in doing this FIRST is to pick the right position for the market.

Bullish?

Buy a call.

Bearish?

Buy a put.

Want to bet on volatility?

Buy a straddle.

Know the breakeven points, max gain and loss - before you put the trade on.

There's one thing missing though.

You'll notice all these payoffs happen when the options expire.

What about before they expire?

The Black-Scholes formula gives us the value of an option at any time before expiration.

In the next thread, we'll see how to build the formula in Python.

Here's a preview:

Now, you can apply this framework to your analysis.

Step 1 is understanding risk before you make the trade.

If you want to learn how to do this with Python, grab my 46-Page Ultimate Guide to Pricing Options and Implied Volatility with Python.

https://t.co/uUXgYrCqgx

PyQuant News writes about resources for using Python for quantitative and data analysis.

• Reply to this thread with any questions
• Follow me @pyquantnews for more of these

• RT the tweet below to share this thread with your audience https://t.co/5BrPwf19lc

> In 2012, my first options trade lost $9,000.
>
> 12 months later I was making $1,100 per week trading in my free time.
>
> What changed?
>
> I read 20 books on options and finished a master\u2019s degree.
>
> But what took my game to the next level was Python.
>
> Here\u2019s the code I still use today:
>
> — PyQuant News (@pyquantnews) August 11, 2022

A few people asked me about books for options.

Start here:

• https://t.co/cF5wUTyjNp
• https://t.co/OlMy9zFGL3