Options PnL Heatmap

Abstract

This report accompanies the code in the corresponding GitHub repository. We present a simple tool for visualising European call option profit & loss. The user inputs a stock symbol, expiry date and a number of contracts (as well as a spacing specification for aesthetics of the output), and the output is a heatmap showing PnL across various strike and underlying prices.

Contents

1 Overview of Options Contracts

An options contract is an example of a *derivative* instrument; one that *derives* its value from another, *underlying* instrument. Options contracts can be divided into two categories; **Call Options** and **Put Options**. For a **European Call Option**, the *seller* is *obliged* to provide the stock of a company (the *underlying*) at some predetermined *strike price K*, at some fixed time in the future. The *buyer* of a European Call Option has the *right*, but not the obligation, to *exercise* or *not exercise* the option.

If the price of the stock at the maturity of the contract is more than the strike price K, the buyer should exercise the option. If they buy the stock at the strike price, they can then immediately sell their stock into the market, making a profit. The converse is true if the price of the stock is less than the strike price; the buyer should choose not to exercise the option in this case.

Conversely, for a **European Put Option**, instead the *buyer* of the contract has the *right* to sell the underlying stock at the strike price, and the *seller* of the option has the *obligation* to buy this stock at the strike price should the buyer choose to exercise.

The precise form of the payoff of an options contract is called the **Payoff Function** $g(S)$, where S is the stock price. There are two different forms of options contracts;

- **Path-Independent Options** are defined by a payoff function that only depends on the value of the underlying security at maturity, and is independent of how the security gets to its final price. European Calls and Puts are examples of Path-Independent Options.

- **Path-Dependent Options** are defined by a payoff function that depends on the entire path the security takes before the option expires. American Options are examples of Path-Dependent Options; these are similar to European Options, with the added twist that they can be exercised at any time before the expiry of the contract.

Let $S(t)$ be the price of a security underlying a European call option. The price of a European call is denoted $\mathcal{C}(t) \equiv \mathcal{C}(t, S(t))$, and gives the owner (ie, the buyer) of the call the *right* to buy the security at a future time $T > t$ for the strike price K. At maturity $t = T$, the value of the call (for the buyer) is

$$
\mathcal{C}(T, S(T)) = \begin{cases} S(T) - K, & S(T) > K \\ 0, & S(T) < K \end{cases}
$$
$$
\equiv g(S). \tag{1.1}
$$

Notice that the lowest value $\mathcal{C}$ can take is 0; the above assumes that the buyer of the call does not exercise the option if $S(T) < K$. The above also ignores the premium paid to the issuer.

2   Methodology

2.1   Profit & Loss

In this project, our goal is to visualise the profit & loss the buyer of any given European call option is exposed to. We use a slight twist on the above value of the call; we define the profit & loss at maturity to be given by

$$\text{PnL} := 100\mathcal{N}(S(T) - \mathcal{P} - K), \tag{2.1}$$

where $\mathcal{N}$ is the number of contracts, $S(T)$ is the price of the underlying stock at maturity, K is the strike price and $\mathcal{P}$ is the *premium*; the price paid to the issuer to enter the options contract. We assume that each contract provides exposure to 100 shares of the underlying. Notice that we have multiple scenarios;

$$\text{PnL} \begin{cases} > 0 & \text{If } S(T) - \mathcal{P} > K, \\ = 0 & \text{If } S(T) - \mathcal{P} = K, \\ < 0 & \text{If } S(T) - \mathcal{P} < K. \end{cases} \tag{2.2}$$

It is the PnL that the tool visualises, not the value defined in equation 1.1.
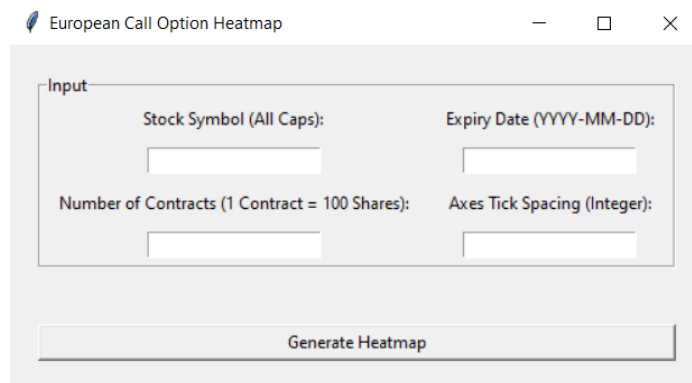
2.2   Overview of the code

The code begins by defining some functions used for saving valid data entered by the user and for producing error messages if the user enters invalid data (for instance, if they enter a string into a field that takes an integer).

We then use the tkinter package to build a GUI containing a number of widgets - four text input fields requesting a stock symbol, expiry date, number of contracts and a tick spacing for aesthetics of the output plot. We also have a button ("generate heatmap") that the user clicks once input is complete.

After the user-inputted data is saved to a list, the corresponding options chain is downloaded from Yahoo! Finance and separated into calls and puts. A heatmap is then generated with the available strike prices on the y axis, various potential underlying stock prices on the x axis, and the heat being the profit & loss calculated via equation 2.1. A table appears above the heatmap reminding the user of the corresponding stock, expiry date and number of contracts.
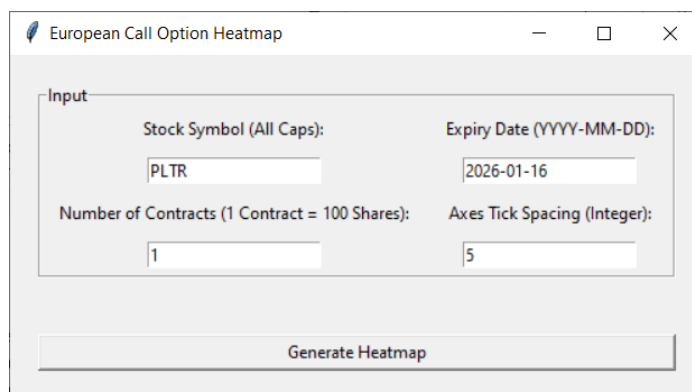
3   Example

Here we demonstrate the code in action. Upon running the python script, the window in figure 4 appears. The window takes a stock symbol, expiry date, number of contracts and axes tick spacing as inputs.

Figure 1: The empty window that appears upon running the python script.

If the user inputs anything incorrectly, a number of error messages appear prompting the user to correct their errors, which can be seen in appendix A. A correctly filled window can be seen in figure 2.

Figure 2: An example of a correctly filled window. In this case, the user is interested in calls where the underlying stock is Palantir, the expiry date is $2026-01-16$, and they want to buy 1 contract.

Upon clicking the "Generate Heatmap" button, a heatmap is generated showing profit & loss across various underlying prices and the available strike prices. The heatmap generated by the inputs in figure 2 is shown in figure 3.
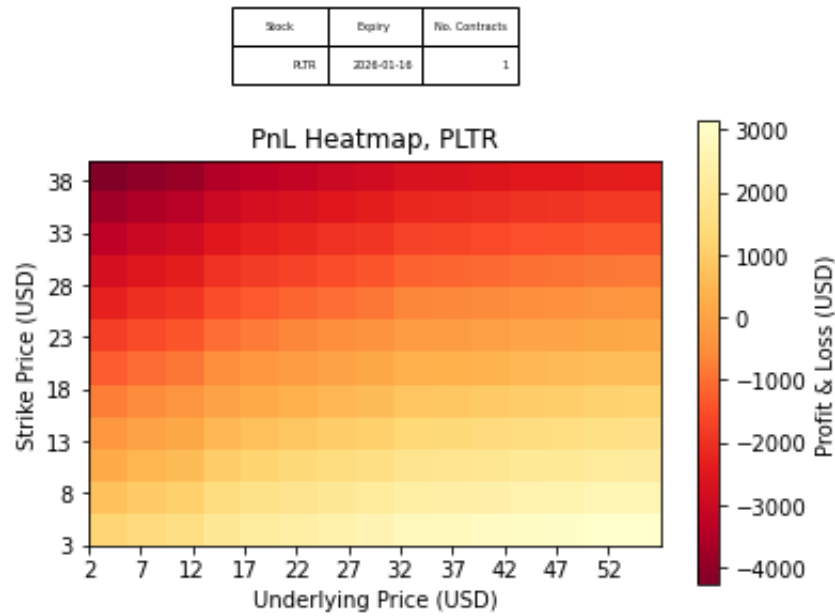
Stock	Expiry	No. Contracts
PLTR	2026-01-16	1

Figure 3: The heatmap generated by the inputs shown in figure 2.

A   Error Messages

If the user inputs anything invalid into the window (for example, they input 1.5 into the number of contracts or a string into the tick spacing), an error message will appear. One can find these error messages below.
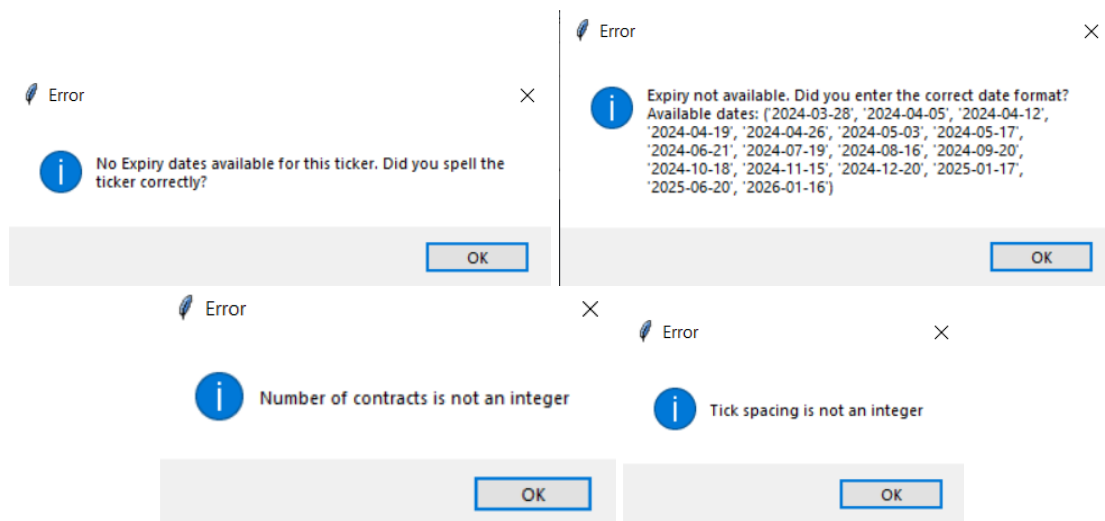
Figure 4: The four possible error messages arising from entering invalid data into the fields. Clockwise starting from top left, these are the results of: entering anything into the stock symbol field that cannot be found in Yahoo! Finance's options database, entering an invalid expiry (whether that be an incorrectly formatted date or an unavailable date), entering a non-integer into the "Axes Tick Spacing" field, and entering a non-integer into the "Number of Contracts" field.