Sofien Kaabar, CFA Follow

Nov 11, 2021 · 7 min read · · Listen

Save

A Profitable Trading Strategy on the Keltner Channel.

Coding a New Trading Strategy in Python.

www.pxfuel.com

We do not have to stick to theoretical and classical strategies that are presented with indicators. The default version is rarely if ever profitable or even useful which is why we

I have just released a new book after the success of my previous one "*The Book of Trading Strategies*". It features advanced trend-following indicators and strategies with a **GitHub** page dedicated to the continuously updated code. Also, this book features the original colors after having optimized for printing costs. If you feel that this interests you, feel free to visit the below Amazon link, or if you prefer to buy the PDF version, you could contact me on **LinkedIn**.

Trend Following Strategies in Python: How to Use Indicators to Follow the Trend.

Amazon.com: Trend Following Strategies in Python: How to Use Indicators to Follow the Trend.: 9798756939620: Kaabar...

www.amazon.com

The Keltner Channel

The Keltner channel is a volatility-based technical indicator that resembles the Bollinger bands, only it uses an exponential moving average as the mean calculation and the average true range as a volatility proxy. First of all, we need to define and understand two concepts:

- **Exponential moving averages.**
- **The average true range.**

Moving averages help us confirm and ride the trend. They are the most known technical indicator and this is because of their simplicity and their proven track record of adding value to the analyses. We can use them to find support and resistance levels, stops and targets, and to understand the underlying trend. This versatility makes them an indispensable tool in our trading arsenal.

```
def adder(Data, times):

    for i in range(1, times + 1):

        new = np.zeros((len(Data), 1), dtype = float)
        Data = np.append(Data, new, axis = 1)

    return Data

def deleter(Data, index, times):

    for i in range(1, times + 1):

        Data = np.delete(Data, index, axis = 1)

    return Data

def jump(Data, jump):

    Data = Data[jump:, ]

    return Data

def ma(Data, lookback, close, where):

    Data = adder(Data, 1)

    for i in range(len(Data)):

     try:
       Data[i, where] = (Data[i - lookback + 1:i + 1, close].mean())

            except IndexError:
                pass

    # Cleaning
    Data = jump(Data, lookback)

    return Data

def ema(Data, alpha, lookback, what, where):

    alpha = alpha / (lookback + 1.0)
    beta  = 1 - alpha
```


```
    Data[lookback + 1, where] = (Data[lookback + 1, what] * alpha) +
(Data[lookback, where] * beta)

    # Calculating the rest of EMA
    for i in range(lookback + 2, len(Data)):
            try:
                Data[i, where] = (Data[i, what] * alpha) + (Data[i -
1, where] * beta)

            except IndexError:
                pass

    return Data
```

How about the average true range? Although it is considered as a lagging indicator, it gives some insights as to where volatility is now and where has it been last period (day, week, month, etc.).

First, we should understand how the **True Range** is calculated (the ATR is just the average of that calculation). Consider an OHLC data composed of an timely arrange Open, High, Low, and Close prices. For each time period (bar), the true range is simply the greatest of the three price differences:

- **High — Low**
- **High — Previous close**
- **Previous close — Low**

Once we have got the maximum out of the above three, we simply take a smoothed average of n periods of the true ranges to get the average true range.

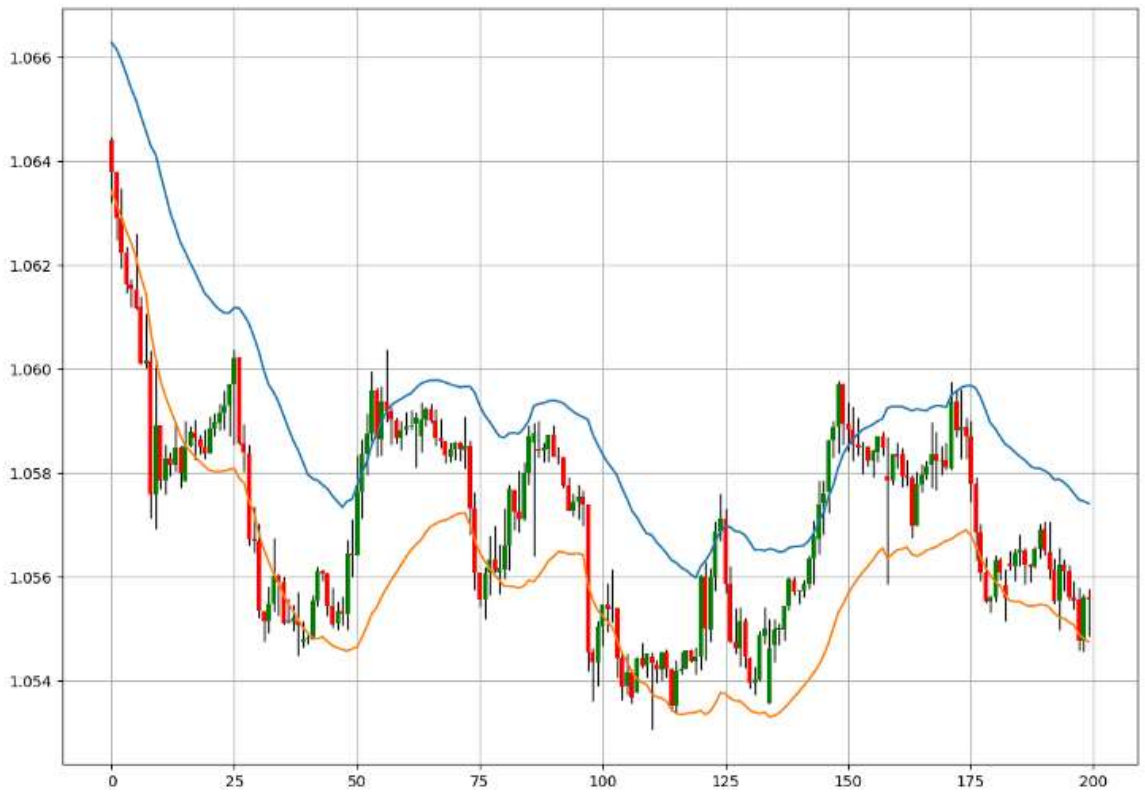
```
def atr(Data, lookback, high, low, close, where, genre = 'Smoothed'):

    # Adding the required columns
    Data = adder(Data, 2)
```


```
                                    abs(Data[i, high] - Data[i - 1,
close]),
                                    abs(Data[i, low]  - Data[i - 1,
close]))

            except ValueError:
                pass

        Data[0, where] = 0

        if genre == 'Smoothed':

            # Average True Range Calculation
            Data = ema(Data, 2, (lookback * 2) - 1, where, where + 1)

        if genre == 'Simple':

            # Average True Range Calculation
            Data = ma(Data, lookback, where, where + 1)

        # Cleaning
        Data = deleter(Data, where, 1)
        Data = jump(Data, lookback)

        return Data
```

Now, we calculate the Keltner channel using an exponential moving average with the ATR of the price. Here is the formula:

$$Upper\ Band = Exponential\ Moving\ Average + (Constant\ .ATR)$$

$$Lower\ Band = Exponential\ Moving\ Average - (Constant\ .ATR)$$



Hourly EURCHF values with the 20-period Keltner channel using a multiplier of 2.

With the Python code to output the Keltner Channel:

```
def keltner_channel(Data, ma_lookback, multiplier, what, where):

    # Adding a few columns
    Data = adder(Data, 3)

    Data = ema(Data, 2, ma_lookback, what, where)

    Data = atr(Data, ma_lookback, 2, 1, 3, where + 1)

    Data[:, where + 2] = Data[:, where] + (Data[:, where + 1] *
multiplier)
    Data[:, where + 3] = Data[:, where] - (Data[:, where + 1] *
multiplier)
```

Hourly USDCHF values with the 20-period Keltner channel using a multiplier of 2.

The idea of the channel is to try to envelop the market price between two boundaries that represent normality. Any breach of the upper or lower barrier could potentially hint to a mean-reversion trade. Although, it is also used the other way around, a breach may signify that a strong trend is happening and that the move will continue.

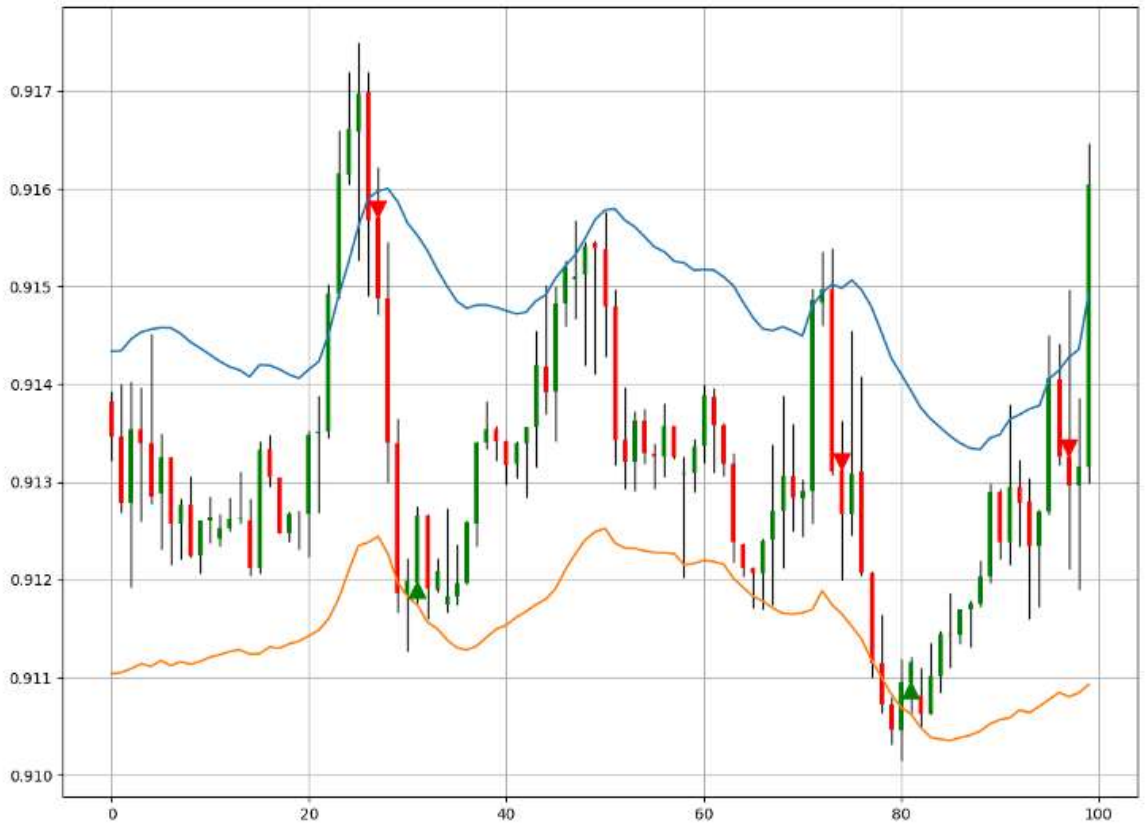
Unfortunately, this gives us two basic conflicting strategies on one indicator with proponents on each side. However, what if we find a third strategy to use the channel?

If you want to see more articles, consider subscribing to my DAILY Newsletter (A ***Free*** Plan is Available) via the below link. It features **my Medium articles, more trading**

Coding & Back-testing the Strategy

The strategy will try to satisfy both Keltner strategists which use the channel either for mean-reversion or trend-following. It is based on the following rules:

- **A long (Buy) signal is generated whenever the market closes higher than the lower Keltner channel after being below it. This serves as a confirmation of reaction and an invalidation of the trend-following trade.**
- **A short (Sell) signal is generated whenever the market closes lower than the upper Keltner channel after being above it. This serves as a confirmation of reaction and an invalidation of the trend-following trade.**



Signal chart.

```
def signal(Data, close_price, upper_keltner_column,
lower_keltner_column, buy_column, sell_column):

    Data = adder(Data, 10)

    for i in range(len(Data)):

        # Bullish Signal
        if Data[i - 1, close_price] < Data[i - 1,
lower_keltner_column] and Data[i, close_price] > Data[i,
lower_keltner_column]:

            Data[i + 1, buy_column] = 1
```

```
    return Data

my_data = keltner_channel(my_data, ma_lookback, multiplier, 3, 4)
my_data = signal(my_data, 3, 4, 5, 6, 7)
```

Signal chart.

If we try a simplistic back-test with the default parameters on the Keltner channel (20, 2). We will find the below results on a sample of major currencies:

Equity curves on the GBPUSD and NZDUSD.

As a comparison, the results on the same currency pairs using the two first strategies slightly underperformed the above strategy which does not prove any superiority but shows that it may be a valid alternative to use the channel.

If you are also interested by more technical indicators and strategies, then my book might interest you:

The Book of Trading Strategies

Amazon.com: The Book of Trading Strategies: 9798532885707: Kaabar, Sofien: Books

Conclusion

Remember to always do your back-tests. You should always believe that other people are **wrong**. My indicators and style of trading may work for me but maybe not for you.

I am a firm believer of not spoon-feeding. I have learnt by doing and not by copying. You should get the idea, the function, the intuition, the conditions of the strategy, and then elaborate (an even better) one yourself so that you back-test and improve it before deciding to take it live or to eliminate it. My choice of not providing specific Back-testing results should lead the reader to explore more herself the strategy and work on it more.

Medium is a hub to many interesting reads. I read a lot of articles before I decided to start writing. Consider joining Medium using my referral link!

To sum up, are the strategies I provide realistic? *Yes*, but only by optimizing the environment (robust algorithm, low costs, honest broker, proper risk management, and order management). Are the strategies provided only for the sole use of trading? ***No**, it is to stimulate brainstorming and getting more trading ideas as we are **all sick of hearing about an oversold RSI as a reason to go short or a resistance being surpassed as a reason to go long. I am trying to introduce a new field called Objective Technical Analysis where we use hard data to judge our techniques rather than rely on outdated classical methods.***