Published in CodeX

Nikhil Adithyan Follow

May 16, 2021 · 11 min read · Listen

Save

Detecting Ranging and Trending Markets with Choppiness Index in Python

An extremely useful indicator to keep yourself away from a pitfall

Introduction

Every technical indicator out there is simply great and unique in its own perspective and usages. But, most of'em never fails to fall in one pitfall which is nothing but the ranging markets. What is a ranging market? Ranging markets are markets that show no trend or momentum but move back and forth between specific high and low price ranges (these markets are also called choppy, sideways, flat markets). Technical indicators are prone to revealing false entry and exit points while markets are ranging. Fortunately, we have a set of indicators that are designed particularly to observe whether a market is ranging or not. Such indicators are called volatility indicators.

There a lot of'em comes under the category of volatility indicators but the one that dominates the game is the Choppiness Index. In this article, we will first understand the concept of the Choppiness Index and how it's being calculated and then code the indicator from scratch in python. We will also discuss how to use the Choppiness Index to detect ranging and trending periods in a ma

Before moving on, if you want to backtest your trading strategies without any coding, there is a solution for it. It is BacktestZone. It is a platform to backtest any number of trading strategies on different types of tradeable assets for free without coding. You can use the tool right away using the link here: https://www.backtestzone.com/

Volatility and ATR

Before jumping on to exploring Choppiness Index, it is essential to have some basic idea on two important concepts which is nothing but Volatility and Average True Range (ATR). Volatility is the measure of the magnitude of price variation or dispersion. Higher the volatility, higher the risk, and vice-versa. People having expertise in this concept will have the possibility to hold a tremendous edge in the market. We have tons of tools to calculate the Volatility of a market but none can achieve a hundred percent accuracy in measuring it but, there are some that have the potential to calculate with more accuracy. One such tool is the Average True Range, shortly known as ATR.

Founded by Wilder Wiles, the Average True Range is a technical indicator that measures

using ATR as it's one of the indicators to track the volatility of a market more accurately. Along with being a lagging indicator, ATR is also a non-directional indicator meaning that the movement of ATR is inversely proportional to the actual movement of the market. To calculate ATR, it is requisite to follow two steps:

- **Calculate True Range (TR):** A True Range of an asset is calculated by taking the greatest values of three price differences which are: market high minus marker low, market high minus previous market close, previous market close minus market low. It can be represented as follows:

```
MAX [ {HIGH - LOW}, {HIGH - P.CLOSE}, {P.CLOSE - LOW} ]

where,
MAX = Maximum values
HIGH = Market High
LOW = Market Low
P.CLOSE = Previous market close
```

- **Calculate ATR:** The calculation for the Average True Range is simple. We just have to take a smoothed average of the previously calculated True Range values for a specified number of periods. The smoothed average is not just any SMA or EMA but an own type of smoothed average created by Wilder Wiles himself but there aren't any restrictions in using other MAs too. In this article, we will be using SMA rather than the custom moving average created by the founder of the indicator to make things simple. The calculation of ATR with a traditional setting of 14 as the number of periods can be represented as follows:

```
ATR 14 = SMA 14 [ TR ]

where,
ATR 14 = 14 Period Average True Range
SMA 14 = 14 Period Simple Moving Average
TR = True Range
```

what volatility and Average True Range are. Let's dive into the main concept of this article, the Choppiness Index.

Choppiness Index

Choppiness Index is a volatility indicator that is used to identify whether a market is ranging or trending. The characteristics of the Choppiness Index are almost similar to ATR. It is a lagging and non-directional indicator whose values rise when the market is bound to consolidate and decreases in value when the market shows high momentum or price actions. The calculation of the Choppiness Index involves two steps:

- **ATR calculation:** In this step, the ATR of an asset is calculated with one as the specified number of periods.
- **Choppiness Index calculation:** To calculate the Choppiness Index with a traditional setting of 14 as the lookback period is calculated by first taking log 10 of the value received by dividing the 14-day total of the previously calculated ATR 1 by the difference between the 14-day highest high and 14-day lowest low. This value is then divided by log 10 of the lookback period and finally multiplied by 100. This might sound confusing but will be easy to understand once you see the representation of the calculation:

```
CI14 = 100 * LOG10 [14D ATR1 SUM/(14D HIGHH - 14D LOWL)] / LOG10(14)

where,
CI14 = 14-day Choppiness Index
14D ATR1 SUM = 14-day sum of ATR with 1 as lookback period
14D HIGHH = 14-day highest high
14D LOWL - 14-day lowest low
```

This concludes our theory part on Choppiness Index. Now let's code the indicator from scratch in Python. Before moving on, a note on disclaimer: This article's sole purpose is to educate people and must be considered as an information piece but not as investment

Our process starts with importing the essential packages into our python environment. Then, we will be pulling the historical stock data of Tesla using an API provided by twelvedata.com (not an affiliate link). After that, we will build the Choppiness Index from scratch.

Step-1: Importing Packages

Importing the required packages into the python environment is a non-skippable step. The essential packages are going to be Pandas to work with data, NumPy to work with arrays and for complex functions, Matplotlib for plotting purposes, and Requests to make API calls.

Python Implementation:

```
import pandas as pd
import requests
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (20, 10)
```

Now that we have imported all the essential packages into our python environment. Let's proceed with pulling the historical data of Tesla with twelvedata.com's powerful stock API.

Step-2: Extracting data from twelvedata.com

In this step, we are going to pull the historical stock data of Tesla using an API endpoint provided by twelvedata.com. Before that, a note on twelvedata.com: Twelve Data is one of the leading market data providers having an enormous amount of API endpoints for all types of market data. It is very easy to interact with the APIs provided by Twelve Data and has one of the best documentation ever. Also, ensure that you have an account on twelvedata.com, only then, you will be able to access your API key (vital element to

```
def get_historical_data(symbol, start_date):
    api_key = 'YOUR API KEY'
    api_url = f'https://api.twelvedata.com/time_series?symbol=
{symbol}&interval=1day&outputsize=5000&apikey={api_key}'
    raw_df = requests.get(api_url).json()
    df =
pd.DataFrame(raw_df['values']).iloc[::-1].set_index('datetime').astype
(float)
    df = df[df.index >= start_date]
    df.index = pd.to_datetime(df.index)
    return df

tsla = get_historical_data('TSLA', '2020-01-01')
tsla
```

Output:

datetime	open	high	low	close	volume
2020-01-02	84.90000	86.14000	84.34200	86.05200	47660500.0
2020-01-03	88.10000	90.80000	87.38400	88.60200	88892500.0
2020-01-06	88.09400	90.31200	88.00000	90.30800	50665000.0
2020-01-07	92.28000	94.32600	90.67200	93.81200	89410500.0
2020-01-08	94.74000	99.69800	93.64600	98.42800	155721500.0
...	...	...	...	...	...
2021-05-10	664.90002	665.02002	627.61011	629.03998	31392417.0
2021-05-11	599.23999	627.09998	595.59998	617.20001	46437200.0
2021-05-12	602.48999	620.40997	586.77002	589.89001	33604000.0
2021-05-13	601.53998	606.46002	559.65002	571.69000	44101000.0

Code Explanation: The first thing we did is to define a function named 'get_historical_data' that takes the stock's symbol ('symbol') and the starting date of the historical data ('start_date') as parameters. Inside the function, we are defining the API key and the URL and stored them into their respective variable. Next, we are extracting the historical data in JSON format using the 'get' function and stored it into the 'raw_df' variable. After doing some processes to clean and format the raw JSON data, we are returning it in the form of a clean Pandas dataframe. Finally, we are calling the created function to pull the historic data of Tesla from the starting of 2020 and stored it into the 'tsla' variable.

Step-3: Choppiness Index Calculation

In this step, we are going to calculate the values of the Choppiness Index with 14 as the lookback period using the Choppiness Index formula we discussed before.

Python Implementation:

```
def get_ci(high, low, close, lookback):
    tr1 = pd.DataFrame(high - low).rename(columns = {0:'tr1'})
    tr2 = pd.DataFrame(abs(high - close.shift(1))).rename(columns =
{0:'tr2'})
    tr3 = pd.DataFrame(abs(low - close.shift(1))).rename(columns =
{0:'tr3'})
    frames = [tr1, tr2, tr3]
    tr = pd.concat(frames, axis = 1, join = 'inner').dropna().max(axis
= 1)
    atr = tr.rolling(1).mean()
    highh = high.rolling(lookback).max()
    lowl = low.rolling(lookback).min()
    ci = 100 * np.log10((atr.rolling(lookback).sum()) / (highh -
lowl)) / np.log10(lookback)
    return ci

tsla['ci_14'] = get_ci(tsla['high'], tsla['low'], tsla['close'], 14)
tsla = tsla.dropna()
tsla
```

datetime	open	high	low	close	volume	ci_14
2020-03-24	95.46000	102.73800	94.80000	101.00000	114476000.0	34.406547
2020-03-25	109.05000	111.40000	102.22200	107.85000	106113500.0	39.103703
2020-03-26	109.47800	112.00000	102.45000	105.63200	86903500.0	43.776124
2020-03-27	101.00000	105.16000	98.80600	102.87200	71887000.0	41.276440
2020-03-30	102.05200	103.33000	98.24600	100.42600	59990500.0	41.623131
...	...	...	...	...	...	...
2021-05-10	664.90002	665.02002	627.61011	629.03998	31392417.0	47.064623
2021-05-11	599.23999	627.09998	595.59998	617.20001	46437200.0	37.315907
2021-05-12	602.48999	620.40997	586.77002	589.89001	33604000.0	36.098058
2021-05-13	601.53998	606.46002	559.65002	571.69000	44101000.0	32.426961
2021-05-14	583.40997	592.87000	570.46002	589.73999	33230600.0	38.064977

Image by Author

Code Explanation: First we are defining a function named ‘get_ci’ that takes the stock’s high price data (‘high’), low price data (‘low’), close price data (‘close’), and the lookback period as parameters.

Inside the function, we are first finding the three differences that are essential for the calculation of True Range. To calculate TR, we are picking the maximum values from those three differences using the ‘max’ function provided by the Pandas package. With the lookback period as 1, we are taking the SMA of TR using the ‘rolling’ and ‘mean’ function to calculate the Average True Index and stored the values into the ‘atr’ variable. Next, we are defining two variables ‘highh’ and ‘lowl’ to store the highest high and lowest low of the stock for a specified lookback period. Now, we are substituting all the calculated values into the formula we discussed before to get the values of the Choppiness Index.

Using Choppiness Index

The values of the Choppiness Index bound between 0 to 100, hence acting as a range-bound oscillator. The closer the values to 100, the higher the choppiness and vice-versa. Usually, two levels are constructed above and below the Choppiness Index plot which is used to identify whether a market is ranging or trending. The above level is usually plotted at a higher threshold of 61.8 and if the values of the Choppiness Index are equal to or above this threshold, then the market is considered to be ranging or consolidating. Likewise, the below level is plotted at a lower threshold of 38.2 and if the Choppiness Index has a reading of or below this threshold, then the market is considered to be trending. The usage of the Choppiness Index can be represented as follows:

```
IF CHOPPINESS INDEX >= 61.8 --> MARKET IS CONSOLIDATING
IF CHOPPINESS INDEX <= 38.2 --> MARKET IS TRENDING
```

Now let's use Choppiness Index to identify ranging and trending market periods of Tesla by plotting the calculated values in python.

Python Implementation:

```
ax1 = plt.subplot2grid((11,1,), (0,0), rowspan = 5, colspan = 1)
ax2 = plt.subplot2grid((11,1,), (6,0), rowspan = 4, colspan = 1)
ax1.plot(tsla['close'], linewidth = 2.5, color = '#2196f3')
ax1.set_title('TSLA CLOSING PRICES')
ax2.plot(tsla['ci_14'], linewidth = 2.5, color = '#fb8c00')
ax2.axhline(38.2, linestyle = '--', linewidth = 1.5, color = 'grey')
ax2.axhline(61.8, linestyle = '--', linewidth = 1.5, color = 'grey')
ax2.set_title('TSLA CHOPPINESS INDEX 14')
plt.show()
```

Output:

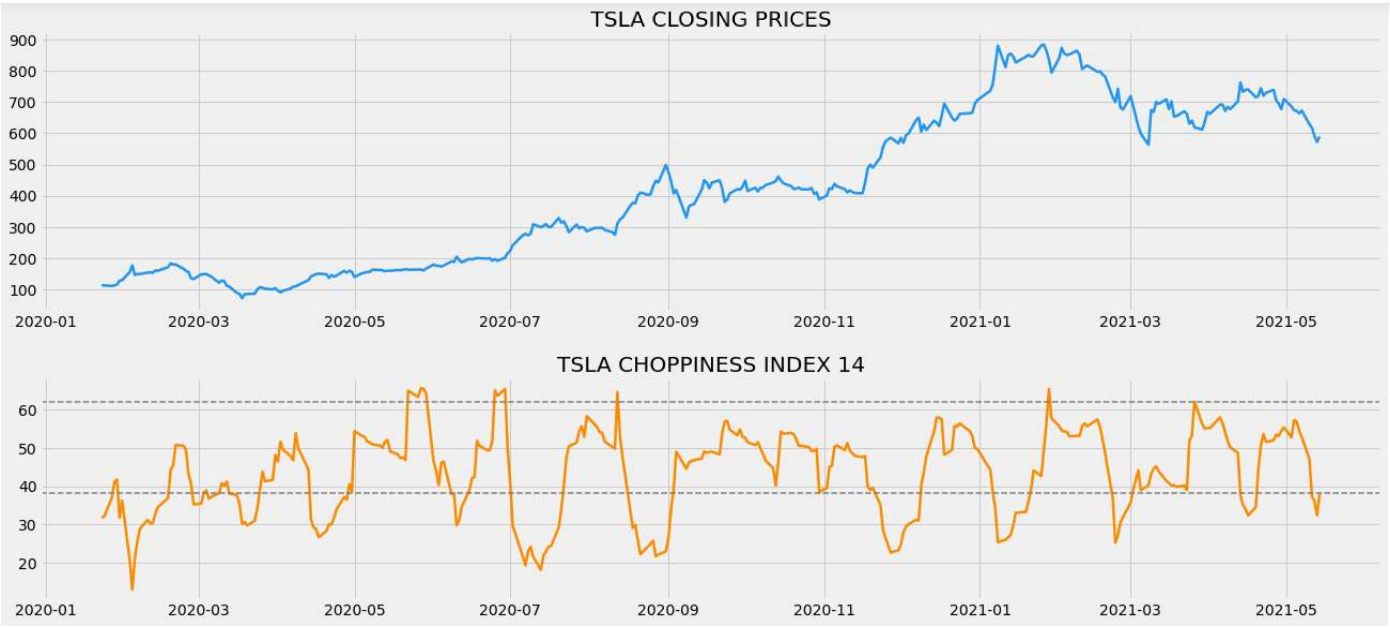

Image by Author

The above chart is separated into two panels: the above panel with the closing price of Tesla and the lower panel with the values of the 14-day Choppiness Index of Tesla. As you can see, there are two lines plotted above and below the Choppiness Index plot which represents the two thresholds used to identify the market movement. Tesla, being a volatile stock with higher price movements, has frequent readings below the lower threshold of 38.2. This means, that Tesla shows huge momentum with higher volatility. We can confirm the readings of the Choppiness Index by seeing the actual price movements parallelly. Likewise, at some places, the Choppiness Index readings exceed the higher threshold of 61.8, representing that the stock is consolidating. This is the traditional and the right way to use the Choppiness Index in the real-world market.

Final Thoughts!

After a long process of theory and coding, we have successfully gained some understanding of what the Choppiness Index is all about and its calculation and usage. It might seem to be a simple indicator but it's very useful while trading and saves you from depleting your capital. Since the Choppiness Index is very lagging, ensure that you're cautious than ever while using it for trading purposes. Also, you should not completely rely just on the Choppiness Index for generating entry and exit points but can be used as a

- **Strategy Optimization:** The Choppiness Index is not just about the higher and lower threshold but it has the potential to be more. So, before entering the real-world market, make sure that you have a robust and optimized trading strategy that uses the Choppiness Index as a filter.
- **Backtesting:** Drawing conclusions by just backtesting the algorithm in one asset or so won't be effective and sometimes can even lead to unexpected turns. The real-world market doesn't work the same every time. In order to make better trades, try backtesting the trading strategy with different assets and modify the strategy if needed.

That's it! Hope you learned something useful from this article. If you forgot to follow any of the coding parts, don't worry. I've provided the full source code at the end of the article. Happy learning!

Full code:

```
import pandas as pd
import requests
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (20, 10)

def get_historical_data(symbol, start_date):
    api_key = 'YOUR API KEY'
    api_url = f'https://api.twelvedata.com/time_series?symbol=
{symbol}&interval=1day&outputsize=5000&apikey={api_key}'
    raw_df = requests.get(api_url).json()
    df =
pd.DataFrame(raw_df['values']).iloc[::-1].set_index('datetime').astype
(float)
    df = df[df.index >= start_date]
    df.index = pd.to_datetime(df.index)
    return df

tsla = get_historical_data('TSLA', '2020-01-01')
```

```
    tr3 = pd.DataFrame(abs(low - close.shift(1))).rename(columns =
{0:'tr3'})
    frames = [tr1, tr2, tr3]
    tr = pd.concat(frames, axis = 1, join = 'inner').dropna().max(axis
= 1)
    atr = tr.rolling(1).mean()
    highh = high.rolling(lookback).max()
    lowl = low.rolling(lookback).min()
    ci = 100 * np.log10((atr.rolling(lookback).sum()) / (highh -
lowl)) / np.log10(lookback)
    return ci

tsla['ci_14'] = get_ci(tsla['high'], tsla['low'], tsla['close'], 14)
tsla = tsla.dropna()
print(tsla)

ax1 = plt.subplot2grid((11,1,), (0,0), rowspan = 5, colspan = 1)
ax2 = plt.subplot2grid((11,1,), (6,0), rowspan = 4, colspan = 1)
ax1.plot(tsla['close'], linewidth = 2.5, color = '#2196f3')
ax1.set_title('TSLA CLOSING PRICES')
ax2.plot(tsla['ci_14'], linewidth = 2.5, color = '#fb8c00')
ax2.axhline(38.2, linestyle = '--', linewidth = 1.5, color = 'grey')
ax2.axhline(61.8, linestyle = '--', linewidth = 1.5, color = 'grey')
ax2.set_title('TSLA CHOPPINESS INDEX 14')
plt.show()
```