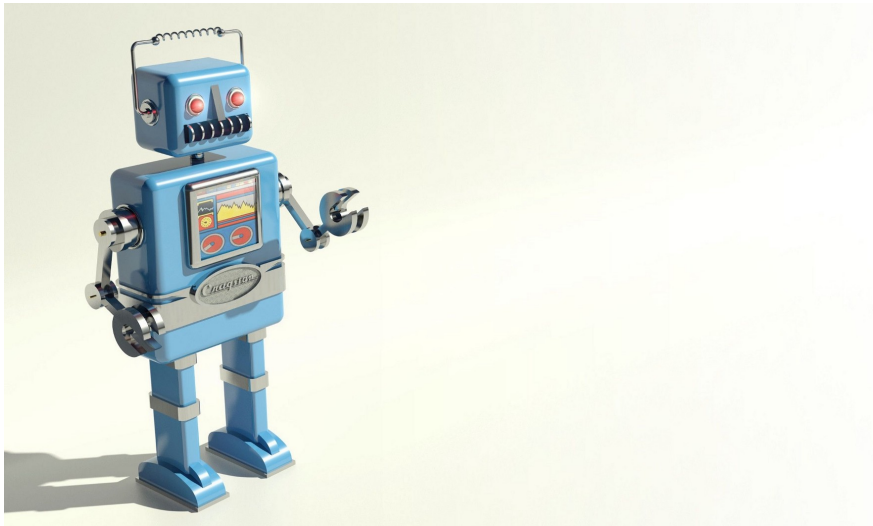
joel degan
I write tutorials that I wish I found, instead of having to write. I like puzzles, games, beer and travel and any combinations of those.
Jun 12, 2017 · 19 min read

Let's write a cryptocurrency bot. (part 1)



> **PLEASE NOTE (1/6/2018)**: This article, and the underlying open source project is being massively rewritten to use the excellent ccxt library and a project called "Coinigy" which both combined allow access to over 100 different exchanges with this system. (yea right!)

> Much of this is to do with the fact that some of what this was originally based on (Bitfinex) has become hostile to persons from the USA and USA persons can't open an account with them.

> That said—the underlying code is probably currently **not working** fully and this article is being rewritten to reflect the changes. You can still go through this, but if you are not a programmer, things are gonna be tough

> COMING SOON: Web configuration, web interface, tutorial videos, Official Docker container AND an AWS AMI.

Want to make your own BTC bot? I'll walk you through the full setup, on to your first execution of an automated trade, and beyond.

— This is a three part article: Part 1, Part 2 and Part 3

Cryptocurrencies

I should not need to tell you but, a couple months ago you could buy the cryptocurrency Etherium for $11, it rapidly went up to $43 (I bought in between those prices) and has now currently gone to over $335 as of June 2017. Those kinds of gains are nearly unbelievable to a traditional investor and yet these are across the board in this space.

Excited yet? So here is a scenario:

So you made a ton of money on cryptocurrencies and have some concerns about shuffling it through your bank because of potential capital gains tax issues. There are places who have a solution for you if you want to be able to use this money for other investments. These places won't make you photograph your license and send it in, just use an email and they provide you with a BTC deposit wallet, demo accounts, API's, then when you are ready, you send money in and it's 'go time', you can trade everything from treasury bonds to Forex using Cryptocurrencies as your base monetary instrument.

But, you say, I am a coder who likes to automate things, surely we can fire up some BTCbot and we can have it just do the work for us, it will make us millions in our sleep, right?

— probably not.

My solution:

I don't want to write a bot and publish it with a single strategy and just say "here, use this", I don't think that is helpful to anyone, I would rather give you the tools and show you how to write strategies yourself, show you how to set up data collection for the strategies and how to implement them in a trading system and see the results.

note: Also, I don't want to create this in a new or arcane language, I want this written in PHP which the most number of people are familiar with and in a framework (Laravel) that is simple to use but powerful enough that

you can create what you need. If you think PHP is just for web pages, read on, this should surprise you.

I like to build systems, I have been working on this post for a while and it represents a good deal of non-derivative custom work. If you have read some of my other tutorials you know that I like to write tutorials that "*I wish that I had found instead of having to to write*", so you are in for a thorough read, with a lot of copy-paste style recipes.

Let's get started.

Steps we are going to take:

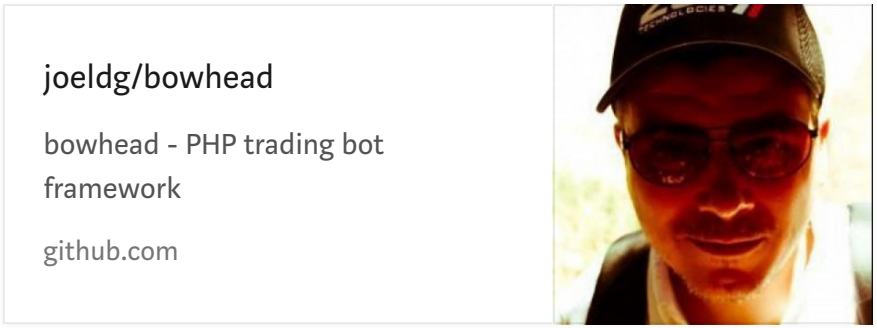
1. Get boilerplate/framework installed.
2. Walk through the core parts of the system, see what is where.
3. Install and configure software we need.
4. Account creation at the brokerages we will be using, setting up the API keys for the scripts.
5. Run tests and examples, lets get it ready.
6. Set up websocket streams to get data.
7. Finding strategies for our automated agents
8. Deep dive into Indicators and Candles available to us.
9. Coding up our first agent.
10. Testing the agent.

11. A few closing words about the risks you are taking.

Get boilerplate/framework installed (Bowhead)

NOTICE: PLEASE READ THE ROADMAP, THIS PROJECT IS IN HUGE FLUX AND I AM IN THE PROCESS OF REWRITING THESES ARTICLES TO MATCH THE CODE.

You can find the repository for the Bowhead boilerplate at it's github repository home here.



Docker UPDATE:

Due to many people having issues with getting a php app running from github, I built out a full Dockerfile to get this set up and running easy for anyone on any platform, it will work the same on Mac, Windows and Linux and can easily be run on Amazon ECS.

You will still need to clone the file, then you will need to have Docker installed. Once that is done, you can use the bowhead/create_docker script or create it via Dockers kitematic app, an alternative to kitematic I like is Portainer by calling it as:

```
docker run -d -p 9000:9000 -v
"/var/run/docker.sock:/var/run/docker.sock"
portainer/portainer
```

and then go to and then go to http://localhost:9000/ to manage your containers.

However this is easy **via command-line**.

```
# create_docker
docker build -t bowhead docker/
docker run --name=bowhead -p 127.0.0.1:8080:8080 bowhead
```

You only need to create the container once, afterwards use the bowhead/start_docker script or

```
docker start bowhead
```

The bowhead API endpoints will be available via localhost:8080/api i.e. http://localhost:8080/api/accounts (these will be explained in part 2)

To ssh into the container use bowhead/ssh_docker or

```
docker exec -it bowhead /bin/bash
cd /var/www/bowhead
ls -alh
# you can now run php artisan from this command line
```

When you make any changes to the scripts use the bowhead/copy_to_docker script

```
chmod 777 storage/logs
chmod 777 bootstrap/cache
docker cp ../bowhead/. bowhead:/var/www/bowhead
docker exec -it bowhead supervisorctl restart all
```

When you are done

```
docker kill bowhead
```

Regular install instructions

> *note: You will need composer installed.*

> *note2: If you are on a Mac, I would strongly suggest the following tools: iTerm2 with the screen command, Sequel Pro and of course PHPStorm*

Bring the project in locally with:

```
git clone https://github.com/joeldg/bowhead.git
cd bowhead
composer update
cp .env.example .env
php artisan migrate
php artisan db:seed
```

```
# now to the other required software
```

```
# If you are on a Mac (this is very roughly)
brew install redis
brew install mysql
brew install php7.1
curl -O http://pear.php.net/go-pear.phar
sudo php -d detect_unicode=0 go-pear.phar
sudo pecl install trader
mkfifo quotes
echo "* * * * * `which php` `pwd`/artisan schedule:run >>
/dev/null 2>&1" | /usr/bin/crontab
```

```
# ################################################
# on ubuntu something like this needs to be done
# updated 06/14/2017
```

```
sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
```

```
sudo apt-get install php7.1 php7.1-opcache php7.1-sqlite3
php7.1-readline php7.1-mysql php7.1-mcrypt php7.1-mbstring
php7.1-json php7.1-intl php7.1-gd php7.1-curl php7.1-bz2
php7.1-bcmath php7.1-cli php7.1-zip php7.1-xml
```

```
sudo apt-get install composer pkg-config php-dompdf php-pear
php-imagick autoconf automake libtool m4
```

```
sudo apt-get install redis-server
sudo apt-get install mysql-server
```

```
sudo apt-get install screen autoconf automake libtool m4
pkg-config
sudo apt-get install php-pear unzip python-pip supervisor
```

```
sudo apt-get install composer pkg-config php-dompdf php-pear
php-imagick autoconf automake libtool m4
```

```
sudo apt-get install php7.1-dev
sudo pecl channel-update pecl.php.net
sudo pecl install trader
```

```
git clone https://github.com/joeldg/bowhead.git
cd bowhead
composer update
cp .env.example .env
php artisan migrate
php artisan db:seed
echo "* * * * * `which php` `pwd`/artisan schedule:run >>
/dev/null 2>&1" | /usr/bin/crontab
```

```
mkfifo quotes
```

```
#see the Dockerfile in docker/ for more help.
```

Let's go over the core parts of the system

Fire up PHPStorm or your favorite editor and load up the project. This system is using the Laravel PHP framework, don't worry you don't need to be familiar with Laravel here, you can learn that on your own time if you want. The primary parts we are concerned with are located in:

- **app/Console/Commands/** which is where all our console commands are located

```
BitfinexWebsocketCommand.php    - Stream market data from
Bitfinex
CoinbaseWebsocketCommand.php    - Stream market data from
GDAX
ExampleForexStrategyCommand.php - Forex example strategy
ExampleStrategyCommand.php      - Our example of a strategy
ExampleUsageCommand.php         - Basic usage examples
GetHistoricalCommand.php        - Pull in historic data from
broker
OandaStreamCommand.php          - Stream market data from
Oanda
```

- **app/Util/** which is all the utility classes that are available

```
Bitfinex.php      - Bitfinex API wrapper
BrokersUtil.php   - Utilities for various brokers
Candles.php       - All 60 TALib candle methods wrapped
Coinbase.php      - GDAX API wrapper
Console.php       - Console color, tables and progress
Indicators.php    - 21 TALib indicators and moving averages.
Oanda.php         - Oanda API wrapper
OneBroker.php     - 1Broker API wrapper
Other.php         - possible indicators, not implemented yet
testStrategy.php  - Here is your test strategy
Whaleclub.php     - Whaleclub API wrapper
```

- **app/Scripts** which has a couple extras and some testing data, these scripts are SKLearn price forecasting scripts taken from a study on beer consumption I thought was really useful, these might be used for market price predictions.

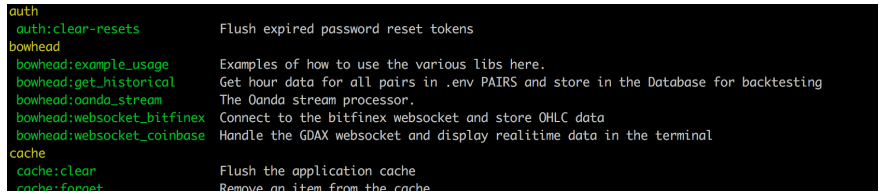
```
close_prediction.py - SKLearn script to predict a closing
price
ohlc-btc.csv        - Sample CSV data, if needed
open_prediction.py  - SKLearn script to predict an opening
price
```

- a python script in the root dir called 'streaming.py' which is part of the Oanda streaming command.
- the **.env-example** file which will need your API keys and be moved to .env

Now, in a terminal in this directory type

```
php artisan
```

You should see something like the following, the part you are interested in is below.



partial output of 'php artisan'

Redis and MySql

If you followed the directions above you should have Redis installed as well as MySql and a few other things.

Redis really does not need any tweaking out of the box, just don't open up ports for it to the outside if you have it on a cloud instance, you don't want people messing with your systems.

MySQL will need a database and a few tables, create your database, I call mine 'bowhead' but you if you want to change this or need to have

prefixed database names then you can change the options in the .env file, here are the defaults in the .env file.

```
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=bowhead
DB_USERNAME=root
DB_PASSWORD=password
```

Once you are satisfied and can connect to your database, then from the terminal you will want to load up our testing data with. Change this command as per your modifications to the .env params above.

```
mysql –u root –p bowhead < app\Script\DBdump.sql
php artisan migrate
php artisan db:seed
```

Open up the database in Sequel Pro and you will the sample data in the bowhead_ohlc (open, high, low, close) table.

API accounts we need in order to set up automated trading.

Note: Where possible I have set up bonuses for you on these links, all sites below offer free accounts which do not require 'verification' and do not require a deposit.

1. Whaleclub the **main** site we want to trade on for this tutorial. They key their market data off of the Bitfinex websocket and match with Oanda streaming data for Forex. This site allows you to trade many instruments and commodities with BTC at up to 20x leverage, Forex up to 222x as well as providing BTC-based binary options. They have a simple easy to understand interface and excellent API. The *API key is found by clicking on your name in the upper-right, and clicking on API. (use DEMO API key to start)*

2. 1Broker the secondary site we want to trade on, they are similar to other BTC-based market makers and have a 'trader follow' system as well that is fairly interesting, particularly to get people following 'you'. *API key is found on the right, just under the email icon, there is a small box with what looks like sliders on it, then click on Access & API Management.*

3. Oanda is where we get our streaming Forex data, you need an account. *API access is found here.*

4. Coinbase/GDAX what used to be called 'Coinbase Exchange' is now called GDAX, I have been automated trading there since they first opened. *API key is found at far upper-right then click on API and create your keys.*

5. Bitfinex you need an account here with an API key so we can get Cryptocurrency quotes. *API keys are found under 'Account' then click on API.*

6. Poloniex is like Bitfinex but supports many alt-coins. *API keys are found here*.

7. TradingView not mandatory, but you will want an account there because all the indicators bowhead uses can be viewed on charts to help you build your strategies.

The reasoning behind this combination is that the Whaleclub and 1Broker API's are rate limited, WC only allows 60 requests per minute, if we want to make sure we have streaming real-time data to work with we need to stream from a BTC brokerage. Same with Forex.

Definitely look around on these sites and see what they have to offer, I've been around the block with a lot of brokers and market maker sites and for BTC, these are all good as of June 2017. For Forex, Oanda is great, but for the purposes here of trading using BTC we just need their streaming Forex data.

Once you get the API keys for these sites, you will want to put them in your .env file in the root bowhead directory. This file is added to the .gitignore so if you like to keep your code in github, then you will be fine.

> NOTE: Start off by using DEMO/TEST API keys, DO NOT use real money API keys with untested trading scripts.

Let's test that we are set up right.

bowhead has a testing script to verify that everything is set up correctly and that you have the right API keys, PHP version and the Trader extension is correctly installed.

```
php artisan bowhead:example_usage
```

This script will stop on any issues that you may have and provide commands to run to fix the issue or links to get API keys you will need.

Let's get data flowing in

We have two things we need to do for data here so we can create an automated trading system that can trade both Crypto and Forex pairs. We will be using this data to trade on BTC market maker sites in real time.

1. Get streaming Forex data coming into our database from Oanda.
2. Get streaming Cryptocurrency data coming into our database from Bitfinex

Note: You should have the screen command installed on either your mac or the cloud instance you are using. Screen is a terminal tool for detaching windows and keeping them running in the background. You can detach a

screen, log off and come back and reattach to it from another location at another time.

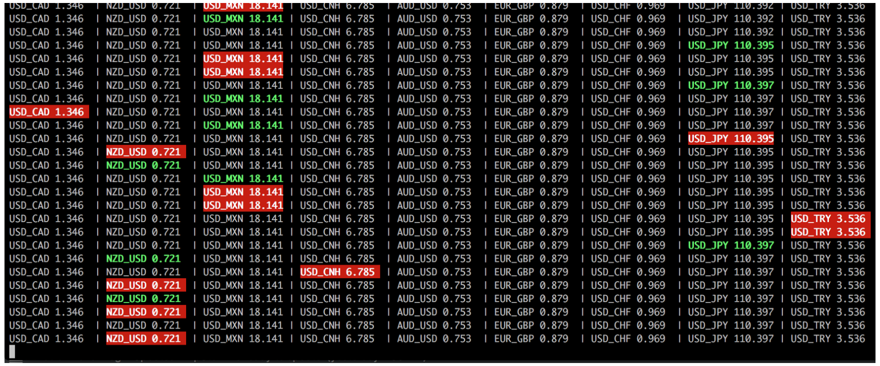
```
~$ screen python streaming.py
*press {control+a+d} to detach*
```

```
~$ screen php artisan bowhead:oanda_stream
*press {control+a+d} to detach*
```

NOTICE: Many users have had issues with newer Oanda accounts, so if you do, use the following by itself.

```
php artisan bowhead:fx_stream
```

Docker has been updated to default to fx_stream method as well.



This is what the Forex streamer looks like if you turn the echo back on.

Now if the Forex markets are open (U.S.A Eastern time, Sunday 5:00pm to Friday 4pm) you will start to see data flowing into the bowhead_ohlc table for the currency pairs that are traded on Whaleclub. The list is in streaming.py and can be modified there. The following pairs are all streaming into your database in real time now. USD_JPY, EUR_USD, AUD_USD, EUR_GBP, USD_CAD, USD_CHF, USD_MXN, USD_TRY, USD_CNH, NZD_USD

Now we have regular Forex data, lets add in the BTC/USD currency pair.

```
~$ screen php artisan bowhead:websocket_bitfinex
*press {control+a+d} to detach*
```

Crypto markets are open 24/7 and you should begin to see current data flowing in immediately.

To see these running processes and reattach to them use `screen -list` and `screen -r`

```
~$ screen -list
There are screens on:
 4604.ttys005.Joels-MacBook-Pro-2 (Detached)
 4636.ttys005.Joels-MacBook-Pro-2 (Detached)
 4652.ttys005.Joels-MacBook-Pro-2 (Detached)
3 Sockets in
/var/folders/bq/79z2kd916hbd39n5bckb5_s00000gn/T/.screen.
```

The numbers on the left are the screen ID's so in this instance you can reattach to the latest (Bitfinex) screen by using the following command.

```
~$ screen -r 4604
```

Your number will be different, screen assigns numbers based on PID.

We are only using screen for the purposes of this tutorial on a local machine, for a server environment we put these on supervisord to

make sure they are always running on our server and if they die, then they are restarted.

This is the supervisord conf I use for this, you may need to change the directory for your user.

```
#~$ more /etc/supervisor/conf.d/crypt.conf
```

```
[program:oanda]
command=/usr/bin/python streaming.py
user=ubuntu
directory=/home/ubuntu/bowhead
startretries=3
stopwaitsecs=10
autostart=true
```

```
[program:o_stream]
command=/usr/bin/php artisan bowhead:oanda_stream
user=ubuntu
directory=/home/ubuntu/bowhead
startretries=3
stopwaitsecs=10
autostart=true
```

```
[program:wsbitfinex]
command=/usr/bin/php artisan bowhead:websocket_bitfinex
directory=/home/ubuntu/bowhead
startretries=3
stopwaitsecs=10
autostart=true
```

You can see what these look like in Supervisor with

```
~$ sudo supervisorctl
o_stream      RUNNING   pid 31644, uptime 1 day, 22:15:24
oanda         RUNNING   pid 31645, uptime 1 day, 22:15:24
wsbitfinex    RUNNING   pid 31646, uptime 1 day, 22:15:24
supervisor> help

default commands (type help <topic>):
=====================================
add    exit      open  reload  restart   start   tail
avail  fg        pid   remove  shutdown  status  update
clear  maintail  quit  reread  signal    stop    version

supervisor>
```

note: Currently bowhead only supports BTC/USD from Bitfinex, I will be adding ETH and LTC in future revisions. You can create an ETH version of this if you want by copying and modifying the BitfinexWebsockCommand.php file to use ETHUSD and renaming the class. You will need to add any new commands class to the `$commands` *array in app/Console/Kernel.php*

Finding strategies

So, we have our boilerplate/framework set up. We have accounts and we have data flowing into our database. we also have our indicator/signals and candles working. Lets jump in and see how to create a very simple strategy.

Now that we see how we can use this, we need strategies and we need to know how to find more strategies. and Quantopian is a great resource for strategies.

for instance, two that I was recently looking at Stocks On The Move and Trading on multiple TA-Lib signals are both interesting, however saying we use TALib methods in bowhead, lets go with the later, additionally this will only be for BTC as Oanda does not return Volume with forex pairs.

You will notice that this strategy uses three signals to determining if a stock (or in our case a pair) is overbought (sell) or underbought (buy).

- Money flow index (mfi)
- Commodity channel index (cci)
- Chande momentum oscillator (cmo)

This is a simple technicals strategy where if all three of these indicators agree then we go the direction they say to go. Here is the core part of the strategy in code.

```
$indicators  = new \Bowhead\Util\Indicators();
$recentData  = $util->getRecentData('BTC/USD');
```

```
$cci = $indicators->cci($instrument, $recentData);
$cmo = $indicators->cmo($instrument, $recentData);
$mfi = $indicators->mfi($instrument, $recentData);
```

```php
/** instrument is overbought, we will short */
if ($cci == -1 && $cmo == -1 && $mfi == -1) {
    $overbought = 1;
}
/** It is underbought, we will go LONG */
if ($cci == 1 && $cmo == 1 && $mfi == 1) {
    $underbought = 1;
}
```

Don't worry about putting this anywhere, this strategy is included in bowhead as a console command

`php artisan bowhead:example_strategy`

*NOTE: **DO NOT** RUN THIS ON YOUR LIVE ACCOUNT UNTIL YOU HAVE TESTED IT, USE YOUR **DEMO API KEY** TO START.*

The output will look like this

```
Joels-MacBook-Pro-2:bowhead joeldg$ php artisan bowhead:example_strategy
PRESS 'q' TO QUIT AND CLOSE ALL POSITIONS


 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
 Signals for BTC/USD:cci:1      cmo:0    mfi:0     overbought underbought
Joels-MacBook-Pro-2:bowhead joeldg$ php artisan bowhead:example_strategy
```

If you would like to see what these look like on a chart, then head over to TradingView and add the indicators, TradingView idea stream is another great place to find strategies and see what other people are doing and you can view the strategies in the source code section of of TradingView.

Bowhead Indicators and Candles

I provide two classes in bowhead for checking signals on data, Candles and Indicators. Each class has a 'all' method which will run all the methods in it's parent class over the data you provide.

To keep things **as simple as possible without sacrifice of functionality** all methods in both of these libraries provide a return as -1, 0 or 1. Where '1' will always be the buy or 'bullish' side and '-1' will always be the sell or 'bearish' side, where applicable. There are a couple which return -100 and 100 as returns, please read the comments above each method and in each class for more info about abnormal return values as there are links to explain what they do and why we use them as well as what they represent and how you can use them in your scripts.

1. Candles.php—The Candles class allCandles() method will check for the presence of 60 specific candles across your dataset. It returns a complex array which will even provide the data point location of the candle and data points around the candle. For purposes of automated scripting, the ['current'] array in the return is the candles that are currently active.

2. Indicators.php —Provides multiple indicators over a dataset, these are all the common technical indicators such as Bollinger bands, RSI and many types of moving averages. These include overlap studies, momentum indicators, volume indicators and volatility indicators. There are no cycle indicators yet. The core methodss are ‘adx’, ’aroonosc’, ’cmo’, ’sar’, ’cci’, ’mfi’, ’obv’, ’stoch’, ’rsi’, ’macd’, ’bollingerBands’, ’atr’ with MA methods of *‘sma’, ’ema’, ’wma’, ’dema’, ’tema’, ’trima’, ’kama’, ’mama’ and ’t3' which can be combined using macdext() fairly dynamically.*

> SMA methods are typically called by themselves as they cannot respond with a buy or sell signal

These two sets of indicators and candles can be combined in many different ways that have been noted in the comments at the top of each class. Combining MA cross overs with Bearish/Bullish candle patterns (which would not be apparent to a moving average) you can pinpoint your entries and exits much better.

Packaging these trader methods in this way provides a lot of flexibility to you to be able to use them very easily and as you notice above, translating a strategy is very simple when you have only buy(1)/hold(0)/sell(-1) signals.

Code it up

So, lets do another quick script that will showcase what we do, this time lets do a Forex bot that trades all the pairs on WC, and it will use

the following technical strategy.

1. Average directional movement index (ADX) is a trend indicator that typically returns a number from 0–100, under 20 it indicates a weak trend, over 50 it indicates a strong trend. Bowhead returns a -1 for under 20 and a 1 for over 50;

2. Two simple moving averages, on period 6 and period 40. Period 6 SMA will follow the price very closely and just smooth out any spikes. An SMA 40 is a much more smoothed average which will cross the period 6 at various points when movements start taking place. The ADX is a check that we are indeed in a trend and not in a ranging (sideways)market.

3. When ADX registers a trend (over 50), and our SMA(40) down-crosses the SMA(6) we can buy as the trend is now moving up.

4. When ADX registers a trend and our SMA (40) up-crosses the SMA(6) we can sell as the trend is now moving down.

Here is what this looks like on TradingView, orange in the bottom is the ADX, the green line is the SMA(6) and the blue line is the SMA(40). You can see where you would most likely ‘want’ to do your trades and ‘low and behold’, we have some line crossings at or near those exact places.

everybody wants some!

Seems kind of complicated? Not when you are working in bowhead. The main thing is we need to get the data off the stack for checking previous and current values, that way you can tell when a moving average has crossed another moving average.

```
$recentData = $util->getRecentData($instrument);

$adx         = $indicators->adx($instrument, $recentData);
$_sma6       = trader_sma($recentData['close'], 6);
$sma6        = array_pop($_sma6);
$prior_sma6  = array_pop($_sma6);
$_sma40      = trader_sma($recentData['close'], 40);
$sma40       = array_pop($_sma40);
$prior_sma40 = array_pop($_sma40);
/** have the lines crossed? */
$down_cross  = (($prior_sma6 <= $sma40 && $sma6 > $sma40) ?
1 : 0);
$up_cross    = (($prior_sma40 <= $sma6 && $sma40 > $sma6) ?
1 : 0);
```

Now you can just

```
if ($adx == 1 && $down_cross) {
    $buy = 1;
}
if ($adx == 1 && $up_cross) {
    $sell = 1;
}
```

Testing it

Okay, so I provided this as the following file, ADX will spit errors (-9) without at least 21 data points, so keep that in mind.

```
app/Console/Commands/ExampleForexStrategyCommand.php
```

```
~$ php artisan bowhead:example_forex_strategy
```

This is what it looks like.

Closing words and a note about risk

So now you can find strategies and quickly build your own scripts to trade cryptocurrencies via technical indicators and candle patterns. The sky is the limit.

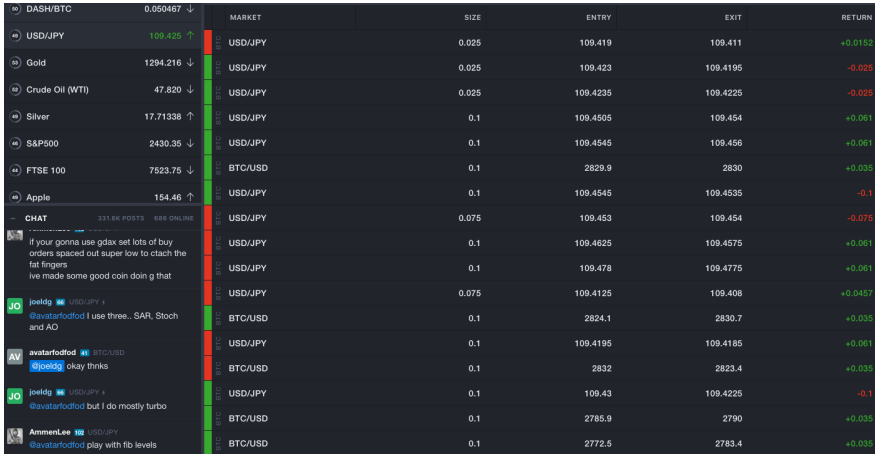
Because this is within the Laravel framework, you can create web pages to manage your automated trading, easily create strategies using web-based tools. You can use the queues and jobs system to have strategy ‘workers’ (I will be adding this as I update it), broadcasts and so on. Laravel is a great framework.

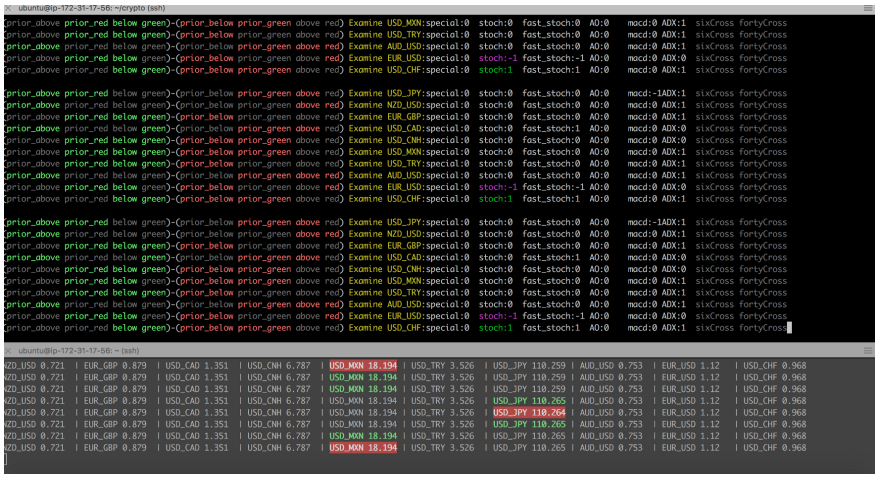
Now to talk about risk.

I would like to point out that there is SUBSTANTIAL risk involved in cryptocurrency trading and you need to make sure you are in demo mode when testing and working out your strategies. This is of paramount importance as I would hate to hear of someone who lost any amount of money because of this.

Myself I am fairly risk tolerant and sometimes it pays off. I use Bowhead to do WC ‘Turbo’ trading (which is Forex Binary options), these are a ‘guess’ if the price will be up or be down in 1 minute and 5 minute contracts. If you guess right then win up to 75% return, if you guess wrong then you lose your entire bet. There are some Forex strategies specifically for Turbo trading that I have had some good luck with. However be aware that most require you are in a trending market. So an indicator like ADX on a longer period is not a bad choice.

Here is an example of one strategy that was working.





This is one of my live systems running Bowhead.

Final note

Part 2 will go over making your bot talk to all the exchanges and even attempt to discern price discrepancies, building real-time GDAX straddle-bot using about five Forex strategies and even setting up Bowhead as an API.

If you notice any errors here or have any issues with the code, please let me know, make a comment here or open an issue in the github repository and I will address it.

— Keep in mind that this project is under active development.

Thanks and enjoy.

CONTINUE TO PART 2

- Joel De Gan

Get social:

If you enjoyed this article, recommend by clicking on that heart icon on the left of this text, and feel free to share it, I work hard on these and feel they are useful

You can also upvote this article on Reddit in r/Bitcoin at this link. https://redd.it/6gpiy6

And of course HN https://news.ycombinator.com/item?id=14538377

joel degan (@joeldg) | Twitter

The latest Tweets from joel degan (@joeldg). I like puzzles, games, beer and travel and any combinations of those...

twitter.com

Joel De Gan | LinkedIn

As Vice President of Technology I architected and built out the REST API and web systems used for RXMG for millions of...

www.linkedin.com

joel degan - Medium

Read writing from joel degan on Medium. I like puzzles, games, beer and travel and any combinations of those. Every day...

medium.com