**Insert Title:**

**Insert subtitle**

## **Jenny Hung (jehung@me.com)**

**Introduction**

In this paper, we implement two trading strategy – the first, based on our FOREX trading plan based on technical analysis, and the second, implemented the neural network with 4 hidden layers based on the same technical indicators as the first strategy. We seek to compare the model performance resulting from both strategies using the combined metrics of portfolio drawdown, investment-period cumulative returns, annualized investment return, as well as Sharpe Ratio. Since no algorithmic trading strategy is ever truly “finished”, we also endeavour to enumerate areas of potential improvement, both in aspects of algorithms as well as daily trade (or position) management policies.

**Rule-Based Trading Strategy**

The first trading strategy we implemented is a rule-based trading strategy, based on our trading plan. The FOREX trading plan considers taking a position using the trade setups of bull pullback, bear rally, bull and bear reversals, as well as high-base breakout and low-base breakdown patterns. The rules associated with each setup is explained below:

Pullback pattern

1. Must have at least one higher high and one higher low
2. Pullback has lasted greater than 3 candles of chosen timeframe
3. Currency pair must be traded above SMA50
4. Traded price EMA9 SMA20 SMA50
5. Price pulled back to between Bollinger Bands (but higher than the lower band)
6. Entry is only made when price is above the parabolic SAR on the chosen time frame

Bear rally pattern

1. Must have at least one lower high and one lower lows
2. Rally has lasted greater than 3 candles of chosen timeframe
3. Currency pair must be traded beelowSMA50
4. Traded price EMA9 SMA20 SMA50
5. Price pulled back to between Bollinger Bands (but lower than the upper band)
6. Entry is only made when price is below the parabolic SAR on the chosen time frame

High-/low-base breakout/breakdown pattern

1. Currency pair is consolidating around EMA9, SMA20 or SMA50
2. Pair is traded above SAM20 and SMA50 (with SMA20SMA50) in a breakout pattern, or traded below SMA20 and SMA50 (with SMA20SMA50) in a breakdown pattern
3. CCI (for breakouts) or CCI (for breakdowns)

Each of the setup in our trading plan uses a similar set of technical indicators, which made the implementation of trading plan on a programmatic basis a task of relative ease. These technical indicators are EMA9, SMA20, SMA50, CCI, Bollinger Bands, and Parabolic SAR.

**Neural Network-Enabled Trading Strategy**

In order to facilitate comparisons of the trading models on an equal footing, we have decided to implement the neural network-based trading model using the same set of technical indicators as the rule-based trading strategy. These technical indicators are EMA9, SMA20, SMA50, CCI, Bollinger Bands, and Parabolic SAR.

**Data Access**

There are two data sources for the trading algorithms. The closing prices, for all currencies, for the past 180 trading days (namely, closing prices on the daily charts for all currencies) was obtained by querying to the quantmod API using the “quantmod” library in R.

However, the quantmod API only returns the closing prices, and gives no details on the OHLC (Open-High-Low-Close) data that is required for computing certain technical indicators, such as CCI and Parabolic SAR. CCI requires the input of high, low, and close of the chosen time frame, whereas the parabolic SAR requires the input of high and low prices of the chosen time frame. The lack of OHLC data results in a lack of data granularity, and this issue has to be resolved before the required data is obtained for the trading algorithms.

The second data source is the Interactive Brokers API, by using the IBrokers library in R. This package will return OHLC data on the daily, as well as weekly, monthly, hourly, minute charts. The use of the Interactive Brokers API therefor resolves the data access issue posted by the lack of data granularity imposed by the quantmod API.

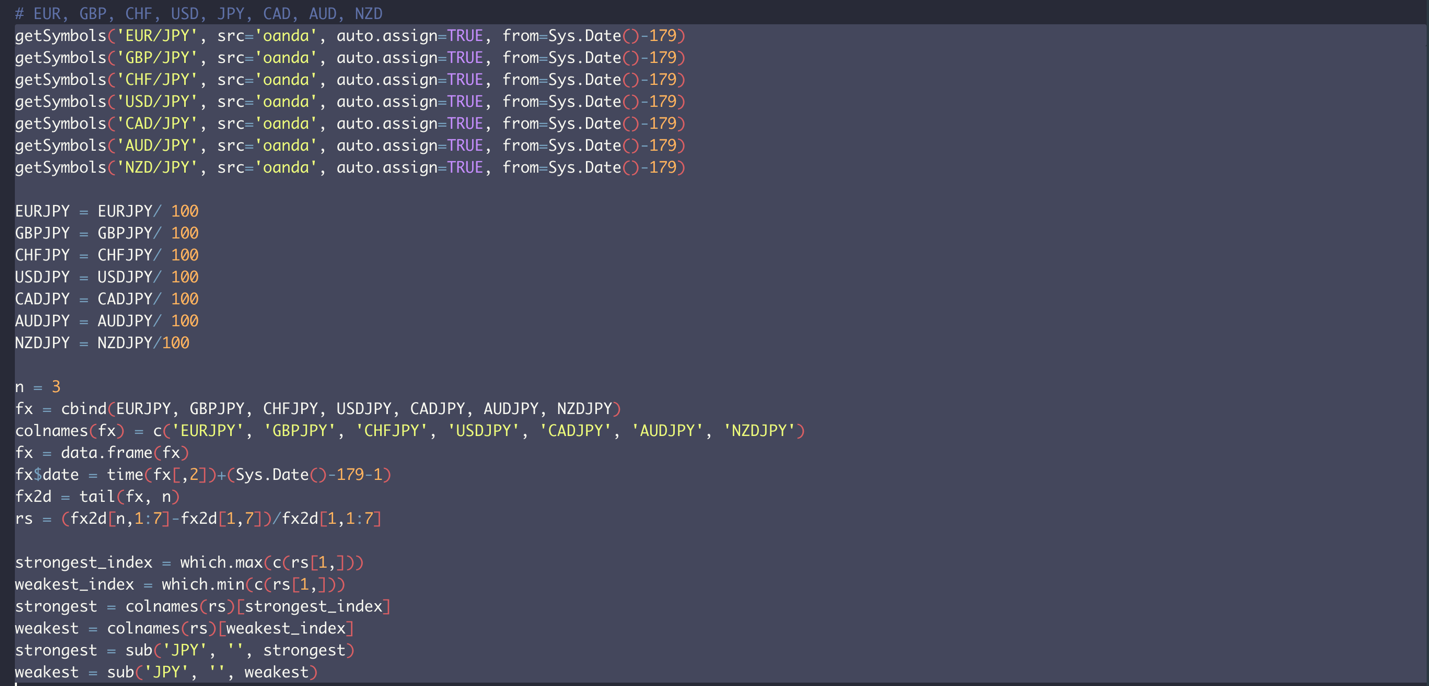
**Conceptual Algorithm Walk-Through**

As discussed earlier, in order to facilitate comparisons of the trading models on an equal footing, we have decided to implement the neural network-based trading model using the same set of technical indicators as the rule-based trading strategy. These technical indicators are EMA9, SMA20, SMA50, CCI, Bollinger Bands, and Parabolic SAR.

1. Relative Strength Computation

Both of the rule-based and the neural network-based trading strategies have the same structure in the algorithms in terms of the methodology with which the algorithm selects a currency pair to trade – we have built in a relative strength calculation across all major currencies (which are EUR, GBP, CHF, USD, JPY, CAD, AUD, NZD). The code to compute relative strength first queries the daily closing data for all major currencies listed above, and subset this data, chooses the strongest as well as the weakest currency using a arbitrary parameter, n. The result of this code is to assemble a strongest-weakest currency pair over the past n-days, where n can be specified by the users. A typical value of has been consumed by the code, but this value can be changed by the user to balance the trade-off of sensitivity and accuracy.

The following code chunk accomplishes this goal:



1. Get trading currency

Once the currency pair is assembled by the algorithm in the above step, the next step is to query the Interactive Brokers API and obtain the name of the trade contract, which in turn is used to get the historical data.

A challenge that encountered is getting the currency trade contract, which potentially raises an error due to the strongest/weakest currency pair sometimes end in recessive pairs, whereas a tradable contact must be a dominant pair. A example of this challenge is when the NZD is the strongest, and CHF is the weakest currency – a NZDCHF contract will result in an code error since NZDCHF is a recessive pair, whereas CHFNZD is the dominant pair and only when using a dominant pair will yield any historical data from the Interactive Broker API.

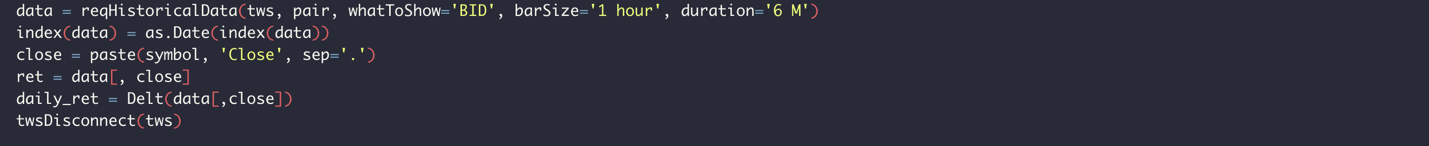
To resolve this potential code error, a tryCatch( ) function in R is utilized.

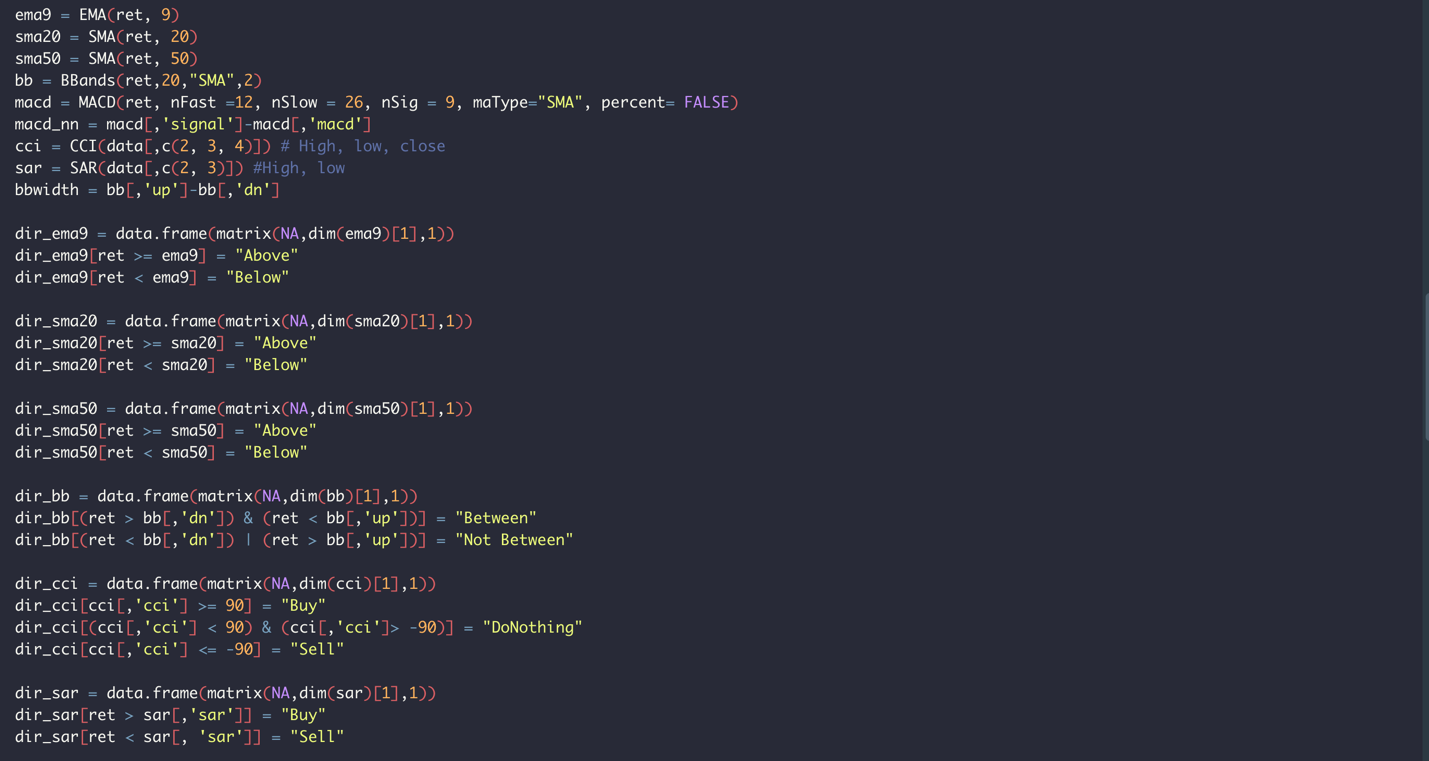
The following code chunk accomplishes this:



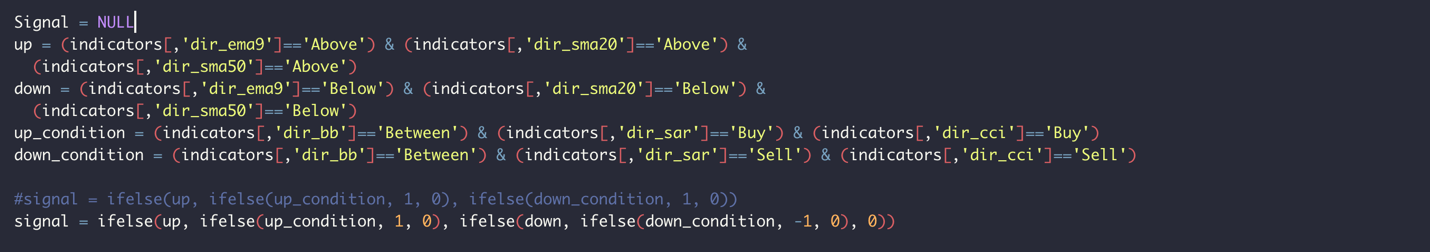
1. Compute binary indicators based on technical indicators

In this process, we have implemented our trading algorithms for speed, by first vectorizing our computations of the technical indicators, and then by comparing the binary indicators using the fast speed bit-wise operations (rather than writing loops). The end result of this stage is an indicator matrix, where each element of the matrix is a signal of a categorical type.





The following code chunk implements the rule-based trading strategy:



The neural network-based trading strategy is implemented as follows:





Our experiments revealed that different inputs are required for the rule-based and neural network-based trading strategy. In particular, binary signals are preferred for the speed of computation for the rule-based strategy, but they failed to carry with them sufficient predictive powers when it comes to neural network-based strategy. As a result, in implementing the neural network, we have used the normalized numerical values of EMA9, SMA20, SMA50, difference between MACD signal and MACD line, as well as the Bollinger Band width as inputs.

1. Performance analysis

At the end of the 180-day period, the trading algorithm then analyzed the performance based on this rule-based strategy by plotting the returns, cumulative returns, drawdowns (using the PerformanceAnalytics library in R), as well as calculating the cumulative returns, annualized returns, and the annualized Sharpe ratio.

****

**Performance Review**

We present the comparative performance analytics under both trading strategies in the chart below. Overall, the rule-based trading strategy is a much simpler model, but yields a cumulative return on an annualized basis of 0.425% over the 25-day test period, whereas the neural network-based trading strategy achieved a cumulative return on an annualized basis of 0.36% over the same period.

As a risk manager, the choice between the two models “as they are” is clear – the rule-based model is simpler and achieves a high return yet at a lower volatility level.

**Dynamic Deployment of Trading Strategy**

Given the success of the rule-based strategy, our natural question for the next step is to enquire

1. Relative Strength Computation

1. Get trading currency

Once the currency pair is assembled by the algorithm in the above step, the next step is to query the Interactive Brokers API and obtain the name of the trade contract, which in turn is used to get the historical data.

A challenge that encountered is getting the currency trade contract, which potentially raises an error due to the strongest/weakest currency pair sometimes end in recessive pairs, whereas a tradable contact must be a dominant pair. An example of this challenge is when the NZD is the strongest, and CHF is the weakest currency – a NZDCHF contract will result in an code error since NZDCHF is a recessive pair, whereas CHFNZD is the dominant pair and only when using a dominant pair will yield any historical data from the Interactive Broker API.

To resolve this potential code error, a tryCatch( ) function in R is utilized.

Code

1. Construct neural networks

In this process, we still have implemented our trading algorithms for speed by vectorizing our computations of the technical indicators. The neural network was constructed using the convenience function nnet( ) in the nnet library in R. We have specified a network with 4 hidden layers.

1. Performance analysis

At the end of the 180-day period, the trading algorithm then analyzed the performance based on this rule-based strategy by plotting the returns, cumulative returns, drawdowns (using the PerformanceAnalytics library in R), as well as calculating the cumulative returns, annualized returns, and the annualized Sharpe ratio.

1. **Iterate by day (for later)**

We conveniently leveraged upon the fact that we have already obtained the 180 days of records through the quantmod API – as a result, for each day we have closing prices of all major currencies, we are able to compute relative strength (step 1), assemble tradable contract(step 2), request historical data through Interactive Brokers(step 2), and deploy rule-based trading strategy (step3).

The daily-by-daily iteration is the only place where the trading algorithm utilizes loops. The loop follows the above workflow. For each day in the period, the trading algorithm will return a signal. We then use a collector to accumulate the daily signal, which was then matched with the daily returns (computation of which was based off the data queried through Interactive Brokers).

**References**

1. Robert H. Shumway and David S. Stoffer, 2010. "Time Series Analysis and Its Applications – with R Examples", Springer Texts in Statistics
2. Berlinger et al., 2015. "Mastering R for Quantitative Finance ", PACKT Publishing
3. Daroczi et al., 2013. "Mastering R for Quantitative Finance ", PACKT Publishing
4. Walter Enders, 2015. "Applied Econometric Time Series ", Wiley
5. Ranjit Kumar Paul, 2000. "Autoregressive Conditional Heteroscedastic (ARCH) Family of Models for Describing Volatility ", I.A.S.R.I Agricultural Statistics
6. Gergana Gerogieva, 2012. "Estimating the Behavioural Equilibrium Exchange Rate of EUR/USD Using a Monetary Model with Labour Productivity", Hanken School of Economics Department of Economics