# Building your own Trading Bot in F#


## Nikhil Barthwal

@nikhilbarthwal

nikhilbarthwal@hotmail.com

www.nikhilbarthwal.com


# Objective

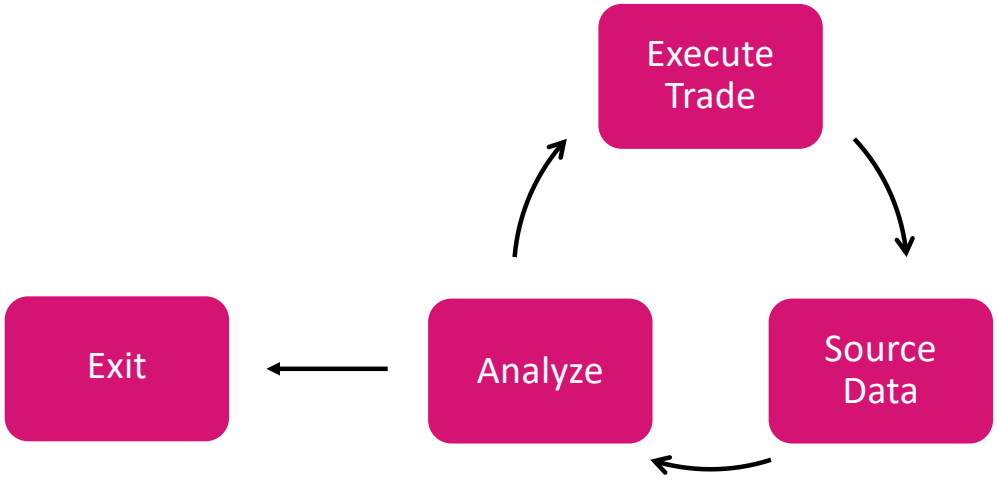
- What is a trading bot? Why do we need it?
- Architecture patterns for Trading bot for its implementation
- Common trading strategies & its implementation

**Disclaimer:**

- Basic knowledge of Trading & Stock market be required to understand some parts.
- Trading can be risky, Use your own judgment.
- All examples are illustration purposes only.

# What is a Trading Bot?

- Program your strategy ahead of time
- Bot would execute it




# Why do we need a Bot to trade?

- Act Rationally without emotions
- Eliminate Greed & Impatience
- Fast executions
- Ability to analyze a lot of data
- Ability to Backtest

# Why use FP to Build a Bot?

- Eliminate bugs, Bugs are costly
- Balance between Productivity & Speed
- Easy to code new trading algorithms or tweak existing ones

CryptoPotato
Everything hot in crypto

Crypto News Margin Trading Guides

## Report: 120,000 BTC ($1.2 Billion) in Jeopardy as a Trading Bot Bugs

Author: George Georgiev • Last Updated Feb 24, 2023 @ 21:40

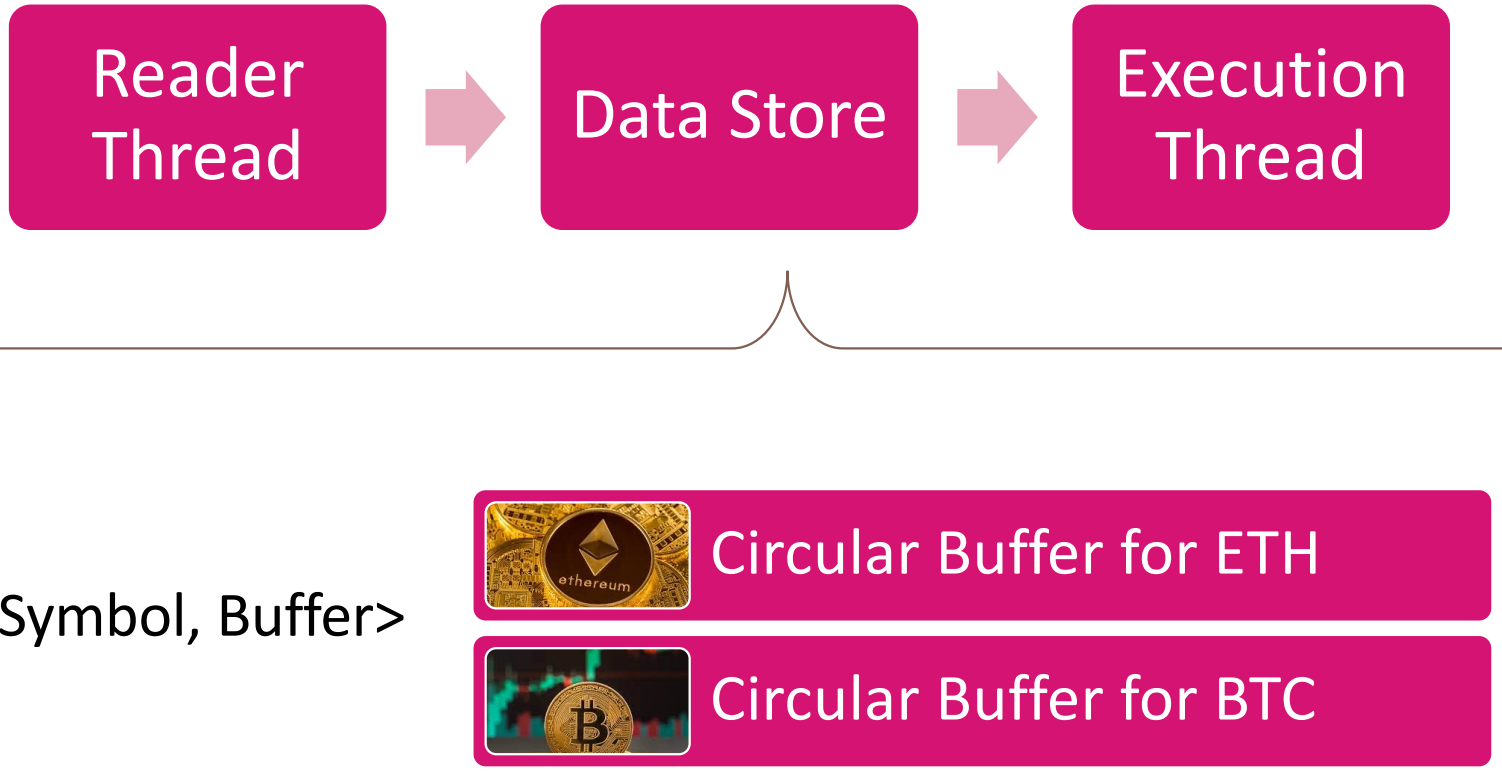
***Some 120,000 Bitcoins are reportedly in jeopardy as a popular Spanish trading bot glitches and customers can't access their funds.***

## Knight Capital Says Trading Glitch Cost It $440 Million

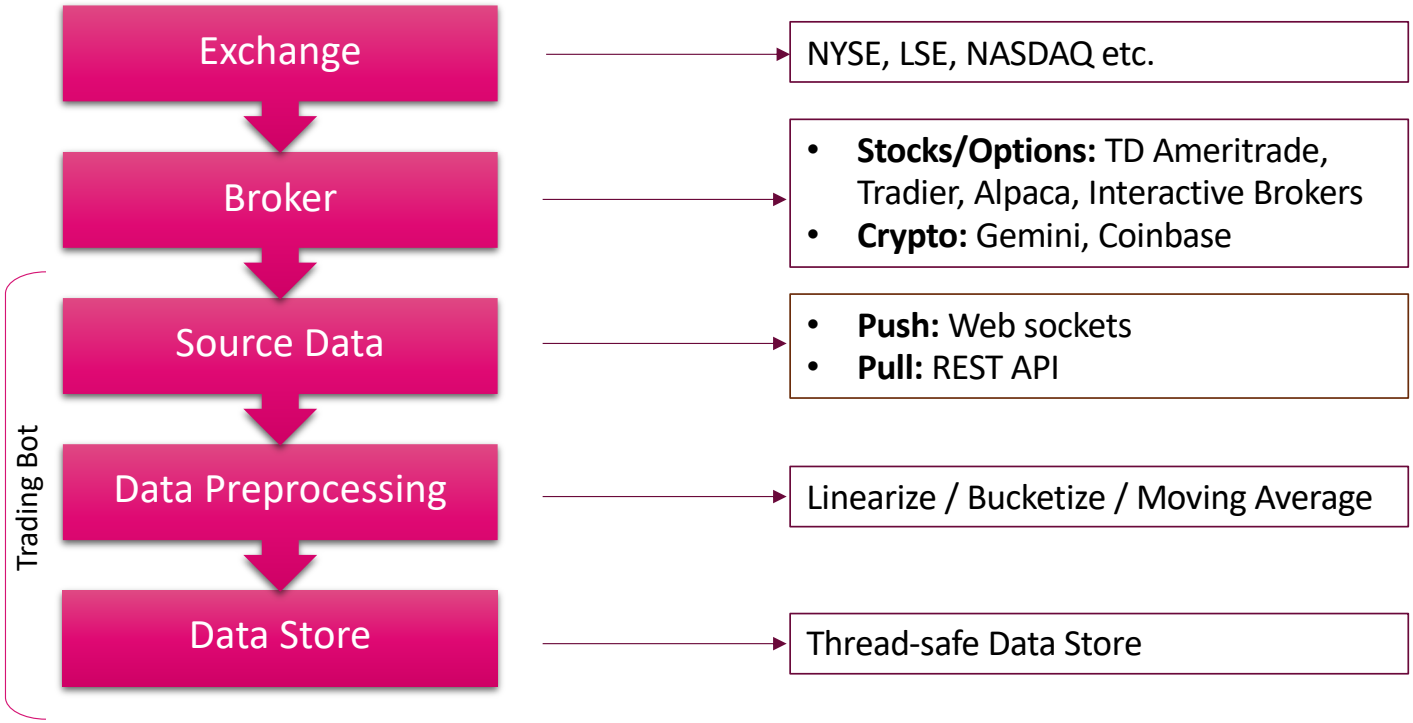
BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.

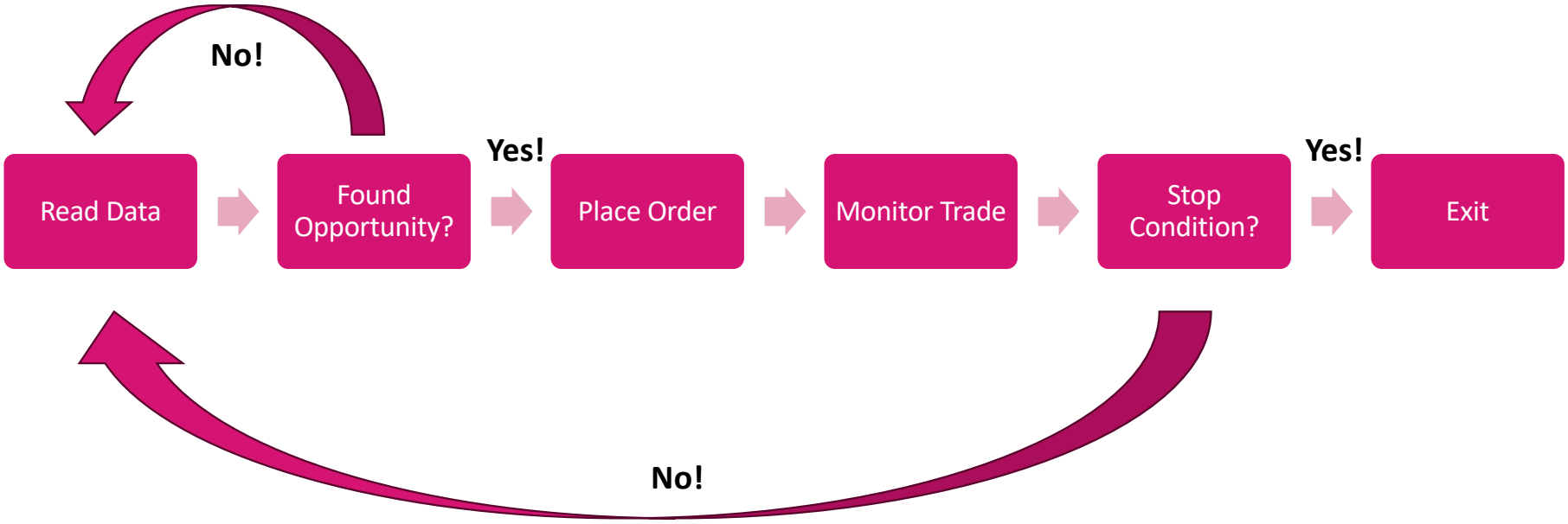
# Reference Architecture



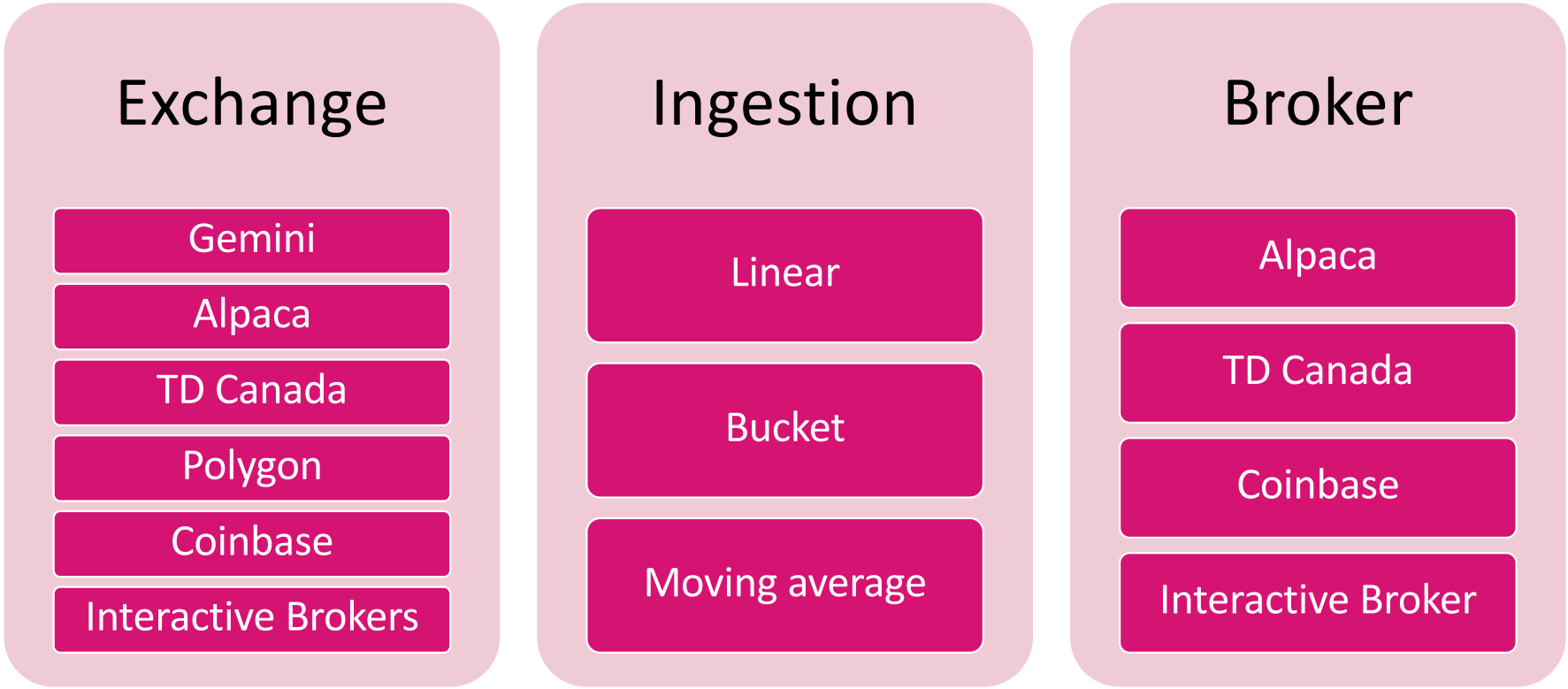
# Reader Thread: Source Data (Push Model)



# Execution Thread: Placing & Monitoring orders



# Abstractions



# Example: Mean Revision Strategy


Explanation: https://medium.com/auquan/mean-reversion-simple-trading-strategies-part-1-a18a87c1196a

# Backtest Strategies

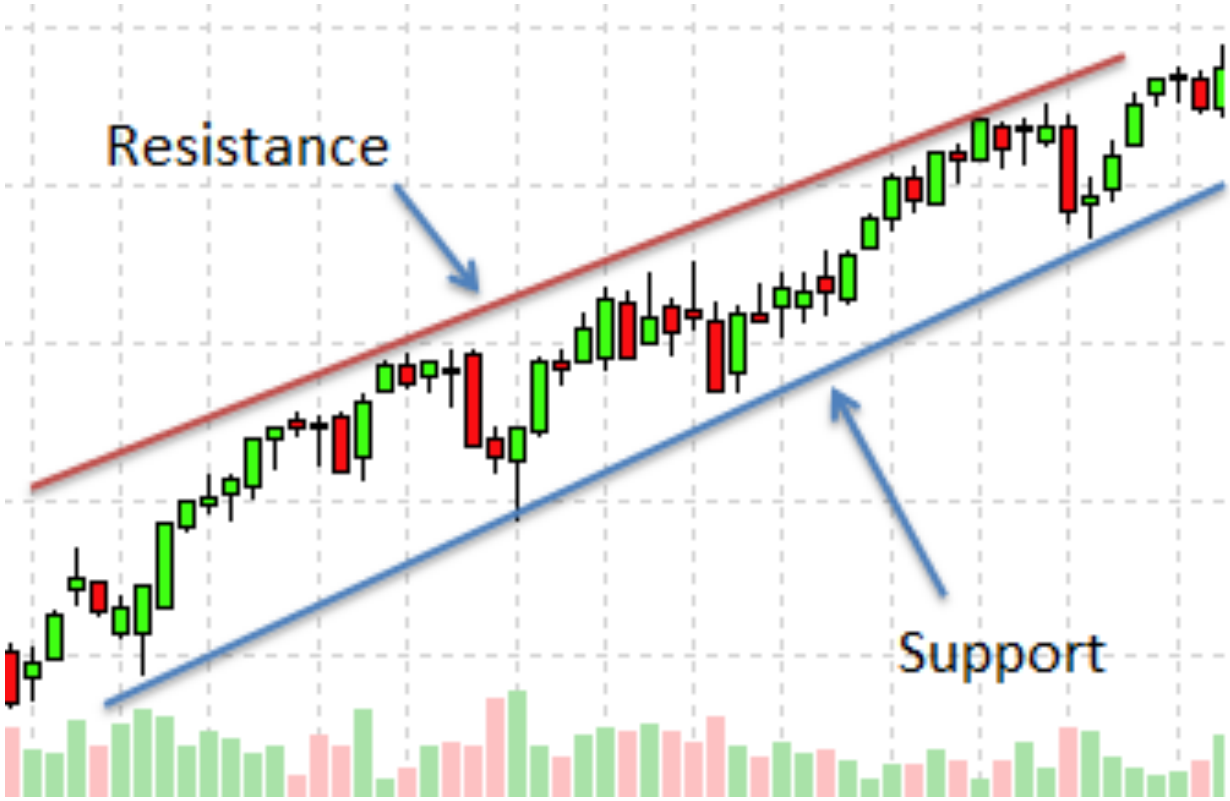
**Use Historical data to how you would have performed in past**

**Good indicator for future**

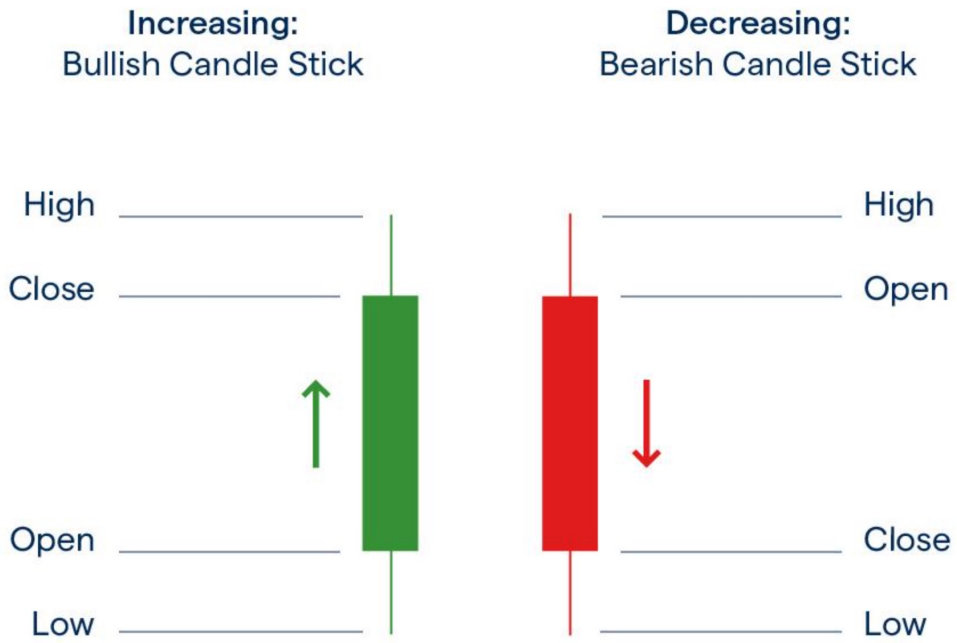
FP Shines in implementing back-testing:

- Compact code
- Code you are trust more!

# Support & Resistance: Definition



# Candles: Definition & Representation




```
type candle = {
    openPrice: float
    closePrice: float
    highPrice: float
    lowPrice: float
    volume: int
}
```

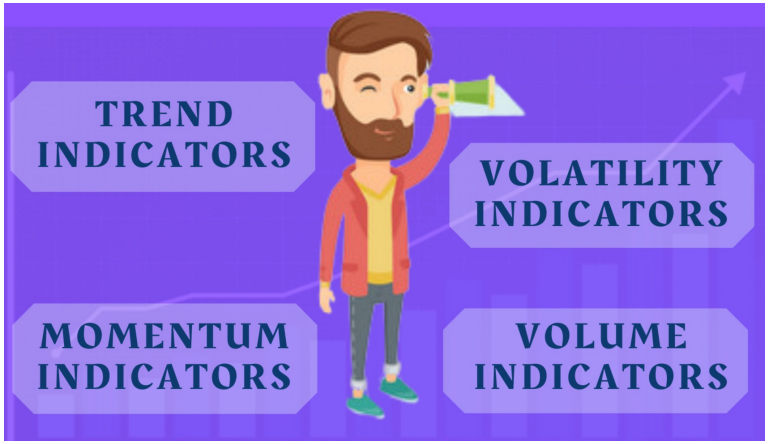
# Support & Resistance: Calculations

```
let data: (int*float)[] = [| ... |]

 let maxima (pos:int): bool =
     (snd data[pos-1] < snd data[pos]) && (snd data[pos] >= snd data[pos+1])

 let minima (pos:int): bool =
     (snd data[pos-1] >= snd data[pos]) && (snd data[pos] < snd data[pos+1])

 let coefficientsLinearLS (f: int -> bool): Vector<float> =
     let points =
         [| 1 .. (data.Length-2) |]
         |> Array.filter f
         |> Array.map (fun x -> data[x])
     let xData = vector (Array.map fst points |> Array.map float)
     let yData = vector (Array.map snd points)
     OrdinaryLeastSquares.Linear.Univariable.coefficient xData yData

 let supportCoefficients = coefficientsLinearLS minima
 let resistanceCoefficients = coefficientsLinearLS maxima
```

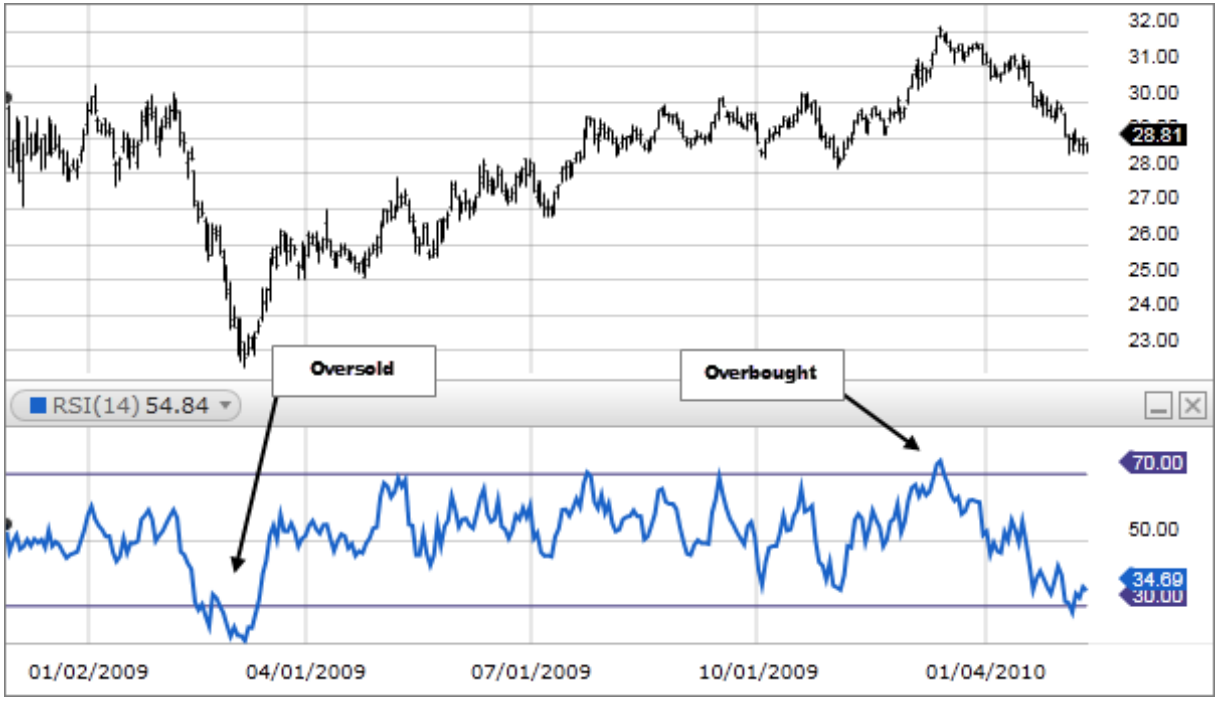
Curve fitting in F#: https://fslab.org/FSharp.Stats/Fitting.html

# Technical Indicators

- Categories: https://www.stockpathshala.com/types-of-technical-indicators/
- Partial List: https://www.barchart.com/education/technical-indicators



# RSI: A Momentum Technical Indicator




Relative Strength Index: https://www.investopedia.com/terms/r/rsi.asp

# RSI: Implementation

```
let rsi (candles: candle[]): float =
    let check (i:int) = candles[i].closePrice > candles[i].closePrice
    let upFilter (i:int) = if (check i) then candles[i].closePrice else 0.0
    let downFilter (i:int) = if not (check i) then 0.0 else candles[i].closePrice
    let apply (f: int -> float) = [1 .. candles.Length] |> List.map f |> List.average
    let rs = (apply upFilter) / (apply downFilter)
    100.0 - (100.0 / (1.0 + rs))
```

Compare it to:

https://github.com/piomin/analyzer/blob/master/analyzer/analyzer-alg/src/main/java/pl/stock/algorithm/core/RSI.java

# A/D Line: A Volume Technical Indicator

Accumulation/Distribution Indicator:
https://www.investopedia.com/terms/a/accumulationdistribution.asp

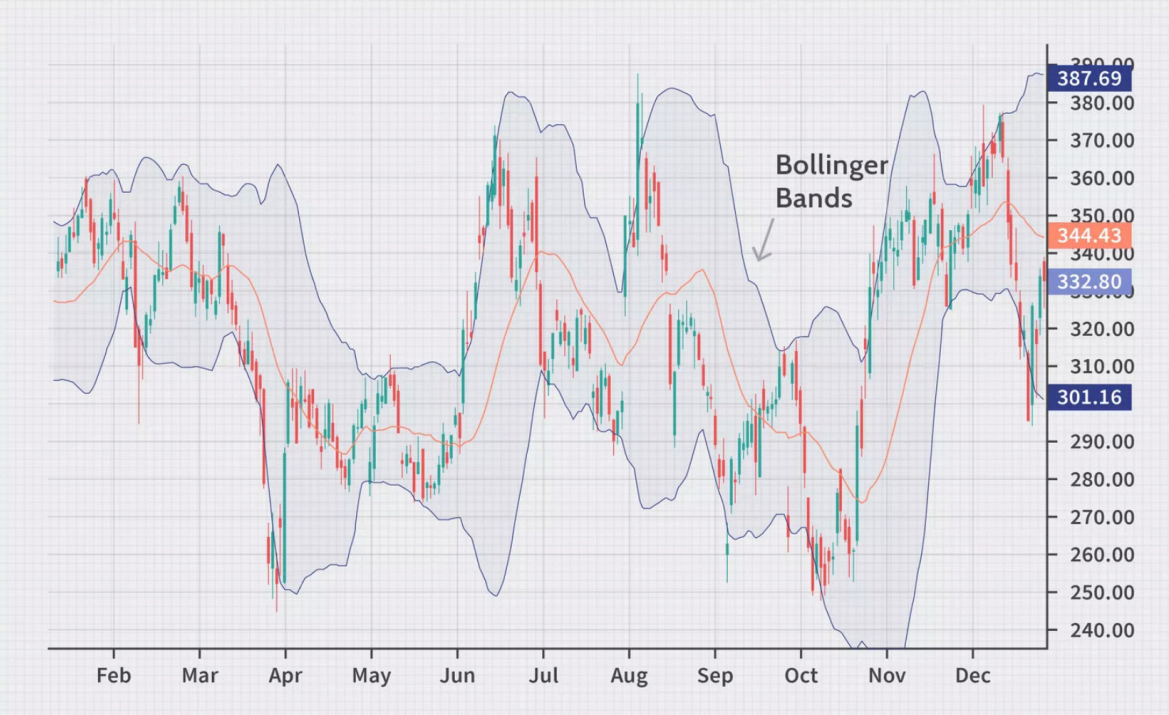
```
let ad_indicator (candles: candle[]): float =
        let clv (c: candle) =
            ((c.closePrice - c.lowPrice) - (c.highPrice - c.closePrice)) /
                (c.highPrice - c.lowPrice)
        candles |> Array.map (fun c -> (clv c) * (float c.volume)) |> Array.sum
```

# Candlestick Patterns

Top 16 patterns: https://www.ig.com/en/trading-strategies/16-candlestick-patterns-every-trader-should-know-180615


```
let hammer (candles: candle[]): bool =
    let bearish (c: candle): bool = c.closePrice < c.openPrice
    let inverted (c: candle): bool =
        (c.highPrice = c.closePrice) && (c.closePrice > c.openPrice)
    (bearish candles[1]) && (bearish candles[2]) && (inverted candles[0])
```

# Bollinger Bands




Reference: https://www.investopedia.com/terms/b/bollingerbands.asp

# Application of Bollinger Bands

```
let bollinger (candles: candle[]) (window: int): bool =
    assert (window > candles.Length)
    let price (pos: int) = (candles[pos].openPrice + candles[pos].closePrice) / 2.0
    let band (pos: int): float * float * float =
        let prices = [| pos .. pos + window |] |> Array.map price
        let mva = Array.average prices
        let dev = Seq.stDev prices
        (mva + dev, mva, mva-dev)
    let touch (pos: int) = let (upper, _, _) = (band pos) in (price pos >= upper)
    let count = [0 .. (candles.Length - 1 - window) ] |> List.filter touch
    let l = (float candles.Length) in count.Length >= int (0.5 * l)
```

# Making Trading Decisions


Various Technical Indicators

Various Strategies

**Analysis uses:**

- Probability & Statistics
- Bayesian Network
- Machine Learning
- Sentiment analysis using NLP
- Technical Indicators

**Trade Order**

# Design Considerations

- Consider putting Stop Losses on trades
- Include terminating conditions
- Monitor trades actively
- Always have a Kill Switch
- Evolve Strategy continuously

# Summary

- Bots can used for trading without emotions or greed
- Functional Programming is a great paradigm for implementing bots
- Always back-test your trading strategies
- Monitor & evolve strategies continuously
- Control risks: Use Kill Switches, Stop loss etc

# Thank you!

**Questions?**


## Nikhil Barthwal

@nikhilbarthwal

nikhilbarthwal@hotmail.com

www.nikhilbarthwal.com