COMP226: Slides 14

BBands overbought/oversold strategy

Rahul Savani

rahul.savani@liverpool.ac.uk

Overview

- **Bollinger Bands** are standard **technical analysis indicator**. They uses moving averages and moving standard deviations

- A **simple mean-reversion type trading strategy** that uses Bollinger Bands

- **Path-dependence**: Stop losses, profit targets, and holding periods as examples of trading strategy constructs that introduce **path dependence**

Bollinger Bands

Parameters:

lookback of moving average and standard deviation	$n > 0$
for multiple of moving standard deviation	$k > 0$

Four components:

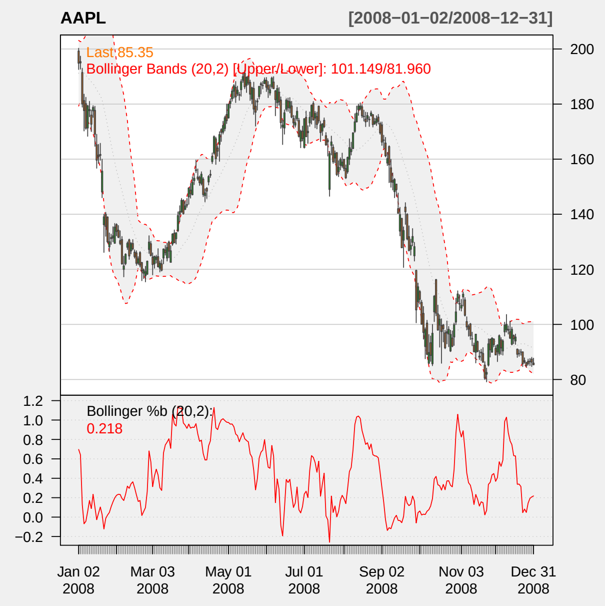
upperBB	$u = s_t + k \cdot \sigma_t$
middleBB	s_t
lowerBB	$l = s_t - k \cdot \sigma_t$
%b	$(x_t - l)/(u - l)$

chartSeries

```
> library(quantmod)
> getSymbols('AAPL')
> taString <- 'addBBands();addBBands(draw="p")'
> chartSeries(AAPL,TA=taString,subset='2008',type='l')
```

This uses the `BBands` function in the package `TTR`

`quantmod` combines `xts` and `TTR` functionality in `chartSeries`


AAPL
[2008−01−02/2008−12−31]
Last 85.35
Bollinger Bands (20,2) [Upper/Lower]: 101.149/81.960
200
180
160
140
120
100
80
Bollinger %b (20,2):
0.218
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
Jan 02 2008
Mar 03 2008
May 01 2008
Jul 01 2008
Sep 02 2008
Nov 03 2008
Dec 31 2008

Example strategy

Overbought/Oversold Strategy

- Long when price is below lower band line
- Short when price is above upper band line

Attempts to trade **corrections** when the market has **"overshot"**:

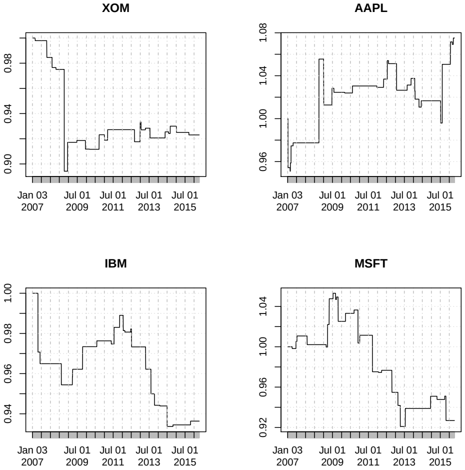
In this sense it is a **mean reversion** type strategy: it bets that prices will move back towards the mean (i.e. the moving average)

Strategy code

We use the same for loop, returns and equity curve calculation as for the copycat strategy, what changes is the **calculation of the position vector**:

```
bbands  <- BBands(prices,n=50,sd=2)
long    <- ifelse(prices<bbands$dn,1,0)
short   <- ifelse(prices>bbands$up,-1,0)
pos     <- long + short
pos     <- lag(pos)
```

Equity curves



Definition

pa·ram·e·ter /pə□ramitər/

Noun:

1. A numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation.
2. A quantity whose value is selected for the particular circumstances and in relation to which other variable quantities may be expressed.

Our strategy's parameters

```
bbands  <- BBands(prices,n=50,sd=2)
```

Two obvious numerical parameters are:

- `n` the lookback, and
- `sd` the standard deviation multiplier

(Non-numeric parameters include the markets we trade on)

Changing `n` and `sd` with have a large impact on this strategy, which we will reflect on in later slides.

More generally we will also return to **optimization** of parameters and the **robustness** of trading strategies later.

Path independence

Definition

A strategy is **path-independent** if

Trading decisions **do not** depend on past trading decisions

- Both strategies we looked at so far were path-independent
- This allowed vectorized computation of positions
- In R, vectorized computation is both **simple** and **efficient**

Entries and exits

- Our BBands mean reversion strategy is path independent because the position on day k is computed without reference to earlier positions

- In terms of **entering** and **exiting** trades (moving between the positions {long, short, flat}), there is symmetry property that results from the path independence:

- Long position: **enter when we cross below lower band line** and **exit when we cross above lower band line**

- For short positions we have a similar symmetry between entering and exiting trades with reference to the upper band

- Note: when exiting a long or short position we may either go to flat or "reverse positions" and enter the "opposite position"

Natural path-dependent variant

Overbought/Oversold Strategy Variant

- Go Long when price **crosses below** lower band line; exit when price **crosses above** moving average
- Go Short when price **crosses above** upper band line; exit when price **crosses below** moving average

Still a **mean reversion** type strategy: bets that prices will move back ~~towards~~ all the way to the mean (i.e. the moving average)

Path dependence in general

- Path independence is actually a serious **limitation**
- Most trading strategies are **path-dependent**
- This means we that the strategy
    - maintains a **state** and
    - **conditions its actions on this state**

Other examples of path dependence

Many common strategy constructions require path dependence:

- Specialized **exit conditions**, e.g.
    - **Holding period**
    - **Profit target**
    - **Stop loss**

Many other examples, e.g., related to **entry conditions** that depend on past performance

Example of holding period

In this example:

- our **state** will encode **how long we have been in a trade**
- we will use a parameter called **hold** (for "holding period")
- we will **exit** trades when we reach our **hold**

In terms of the code implementation, we:

- copy our current position forward to the next period if we have not yet reached our holding period
- if we are in a trade and have hit the holding period we reset the position to flat

Implementation

```
source('run_bbands_hold.R') # contains strategy

pos <- run(prices,n=5,sd=1.5,hold=5) # run strategy
equity  <- getEquityCurve(getLogReturn(prices),pos) # utilities

pdf("equity.pdf")
print(plot(equity,main='Equity curve'))
dev.off()
```

Holding period parameter `hold`

We exit a trade if and only if `hold` periods have passed; implemented by **staying in a trade** if `count` is smaller that `hold`

```r
run <- function(prices,n,sd,hold) {
    lprices <- lag(prices); bbands <- BBands(lprices,n=n,sd=sd)
    pos <- rep(0,length=nrow(prices)) # all zeroes
    for (i in (n+1):nrow(prices)) {
        if (pos[i-1]==0) {
            # compare to prices i-1 to avoid lookahead
            long  <- ifelse(lprices[i-1]<bbands$dn,1,0)
            short <- ifelse(lprices[i-1]>bbands$up,-1,0)
            pos[i] <- long + short
            if (pos[i] != pos[i-1]) count <- 1 # just entered trade
        } else if (count < hold) { # stay in trade
            count <- count + 1; pos[i] <- pos[i-1]
        }
    }
    return(pos)
}
```

Position vector

- The important point is the **path-dependance**

- **Fixed holding period for trades** (5 in the example)

- Position vector comprises 11111, -1 -1 -1 -1 -1, 0; i.e., five days long in a row, or five days short in a row, or a flat day

```
> pos
   [1]  0  0  0  0  0  1  1  1  1  1  0  0  0  1  1  1  1  1  0  0 -1 -1 -1 -1
  [25] -1  0  0  1  1  1  1  1  0  0  0  0  1  1  1  1  1  0  1  1  1  1  1  0
  [49]  1  1  1  1  1  0 -1 -1 -1 -1 -1  0  0  0  0 -1 -1 -1 -1 -1  0  0 -1 -1
  [73] -1 -1 -1  0  0  0 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1  0  0  0  0 -1 -1 -1
  [97] -1 -1  0  1  1  1  1  1  0 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1  0  1
```

- Every run of 11111, or -1 -1 -1 -1 -1 is followed by a 0

Next example - stop loss

- A **stop loss** limits the loss on a particular trade
- We will measure the **simple return of a trade**
- If it is too negative we will exit the trade

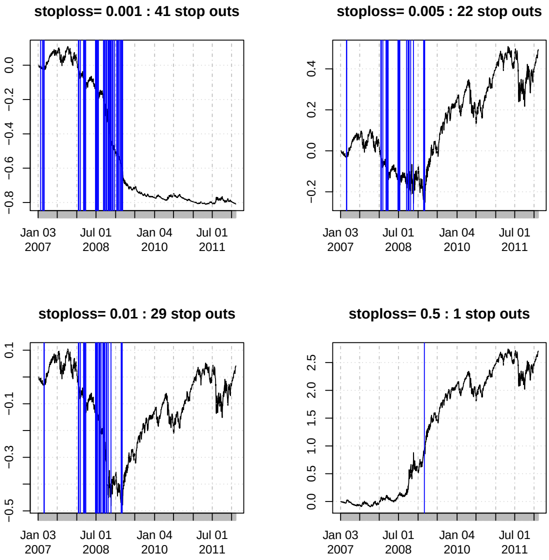
```
getTradeReturn <- function(prices,entry,exit,short=FALSE) {
    prices <- as.numeric(prices)
    if (short)
        prices[entry]/prices[exit] - 1
    else
        prices[exit]/prices[entry] - 1
}
```

```
> prices
           Adjusted
1970-01-02      100
1970-01-03      110
1970-01-04      100
1970-01-05      150
1970-01-06      200
1970-01-07      100
```

```
> getTradeReturn(prices,entry=1,exit=2)
[1] 0.1
> getTradeReturn(prices,entry=1,exit=2,short=T)
[1] -0.09090909
> getTradeReturn(prices,entry=1,exit=4)
[1] 0.5
```

Example

```
run <- function(prices,n,sd,stoploss) {
    lprices <- lag(prices); bbands <- BBands(lprices,n=n,sd=sd)
    pos <- stopOuts <- rep(0,length=nrow(prices)) # all zeroes
    for (i in (n+1):nrow(prices)) {
        if (pos[i-1]==0) { # flat
            long    <- ifelse(lprices[i]<bbands$dn,1,0)
            short   <- ifelse(lprices[i]>bbands$up,-1,0)
            pos[i]  <- long + short
            if (pos[i] != pos[i-1]) entry <- i # remember entry period
        } else {
            ret <- getTradeReturn(lprices,entry,exit=i,isTRUE(pos[entry]<0))
            if (ret > -stoploss) pos[i] <- pos[i-1] # stay in trade
            else stopOuts[i] = 1 # record stopout
        }
    }
    titStr <- paste("stoploss=", stoploss,":",sum(stopOuts),"stop outs")
    plotEquity(prices,pos,stopOuts,titStr); return(pos)
}
```



stoploss= 0.001 : 41 stop outs

stoploss= 0.005 : 22 stop outs

stoploss= 0.01 : 29 stop outs

stoploss= 0.5 : 1 stop outs

Insights from stop loss example

- As me increase the parameter `stoploss`, i.e., make it harder to stopout for a given trade, the number of stopouts generally decreases, **but not** monotonically: **due to path dependence one may increase the "stoploss parameter" and get more stopouts**

- Also it is clear that by **having too small a stoploss parameter**, we **can actually hurt our performance** (we do best with a high parameter of 0.5 in this example); sometimes the market goes against us before going the way we want it to

Profit target

Notice that we can implement a profit target with almost identical code to a stop loss:

Exercise

Convert the previous example into one with a profit target

Finally

We will later introduce a **backtester framework** where we can easily test path-dependent strategies such as the variant of the Bbands mean-reversion strtategy where we exit only when if reach the moving average...