Application of Matlab for Finance
Week 5

Shi, Yining

October 8, 2017

Today's Class

- MA-Crossover Trading Strategy

- Trading Strategy Returns
- Drawdown Curve (Optional)

Obtain Data from External Source: FRED

- Today, we will use Matlab to retrieve data from external sources.
- Download the historical price of S&P500 index for Jan/2008-July/2017.
- The data source is the Federal Reserve Bank of St.Louis (FRED):
  - website: https://research.stlouisfed.org/fred2/
  - code `fred` is the build-in connection from Matlab to FRED
  - ensure the 'Datafeed' package is properly installed
- `yahoo` is another build-in connection in Matlab. However, it is not working since April 2017 when Yahoo changed its API protocol.
- Other connections include `blp` for Bloomberg, `ravenpack` for RavenPack News Analytics, both of which require information on relevant terminals.

Obtain Data from External Source: FRED

```
c = fred; % connect to FRED
ticker = 'SP500'; % define the ticker of the stock
start_date = '2008/01/01';
end_date = '2017/07/31';

data = fetch(c,ticker,start_date,end_date);
close(c) % close the connection
```


- Function `fetch()` retrieves data for a defined stock `ticker` from connection `c` within specified dates
- The information is stored in a *structure* variable called `data`.
- Accessing the *fields* of `data` using dot (.):
  - `data.Data`: the `Data` field of structure variable `data`
  - `data.SeriesID`: the `SeriesID` field of structure variable `data`
- Always remember to close the connection `close(c)`.

Clear Data with Missing Observations

- Time series data may contain missing observations (ie. `NaN`).
- Consider a $(T \times N)$ price matrix with T observations on N stocks.
- We want to delete the days if any stock price on that day is missing.
- `isnan(p)` returns logical output 1 if observation $p(t, n)$ is `NaN`
- `missing = any(isnan(p),2)` returns logical output 1 if any stock on date `t` is `NaN`.
- That is, on row `t`, if `any` $p(t, n)$ `== NaN`, then `missing(t,1)=1`.
- The corresponding row `t` with `missing==1` in the original data is deleted by setting `= [];`
- Note: `any(..,2)` specifies the operation is on the row dimension.


```
%% Clear missing observations
missing = any(isnan(data.Data),2);
data.Data(missing,:)=[];
```

```
%% Read Data & Calculate Returns
spy_t = data.Data(:,1); % read serial timevalue number
% convert date number to string with specified format
spy_t_str = datestr(spy_t,'dd/mm/yyyy')
% read historical price
spy_p = data.Data(:,2);
% calculate continuous return
spy_lnret = tick2ret(spy_p, spy_t, 'continuous');

figure
histogram (spy_lnret, 50)
```


- `data.Data` is a $(T \times 2)$ matrix stores serial datetime number in column 1 and price in column 2.
- The serial datetime number records time in numerical format, which is necessary for time series analysis and figure plot.
- Use function `datestr()` to convert date number into. readable date string, or vice versa with `datenum()`.
- `tick2ret()` calculates the *continuous* return of `spy_p` with time sequence `spy_t`.
- The `histogram` shows that the return distribution is rather symmetric.

Calculate Moving-Average Series

- A moving average at time t with n-day window is the arithmetic average of price from $t-n+1$ to t: $MA\_21_t = \frac{p_t+p_{t-1}+\ldots+p_{t-20}}{21}$


```
% short—term simple moving average 21 days
sma_st = tsmovavg(spy_p, 's', 21, 1);
% long—term simple moving average 126 days
sma_lt = tsmovavg(spy_p, 's', 126,1);
```


- `tsmovavg()` calculates the simple moving average, `'s'`, on the column price vector, `spy_p`, for 21 days and 126 days.
- The last input 1 specifies it is the column matrix dimension.

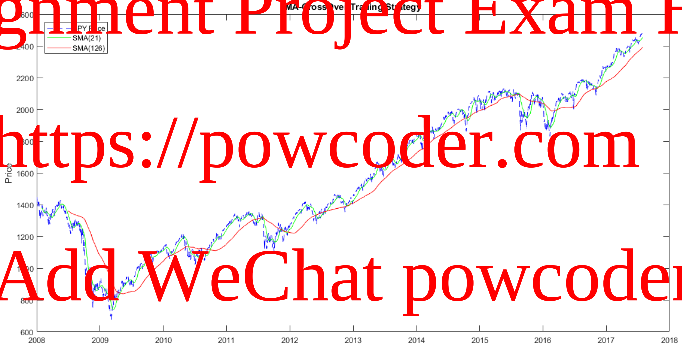
- Other MA calculation includes exponential, `'e'`, triangular, `'t'`.
- In finance, we use working day counting, instead of calender days.
  - 21 days: 1 month
  - 126 days: 6-month
  - 252 days: 1 year

Plot the MA time series

```
clf % clear the previous figure
figure
plot(spy_t,spy_p,'b--',spy_t,sma_st,'g',spy_t,sma_lt,'r')
legend('SPY Price', 'SMA(21)', 'SMA(126)', ...
    'Location','northwest')
dates_limits = [min(spy_t), max(spy_t)];
xlim(dates_limits) % set the x limits
datetick('x') % convert x-ticks into date string
ylabel('Price')
title('MA-CrossOver Trading Strategy')
```

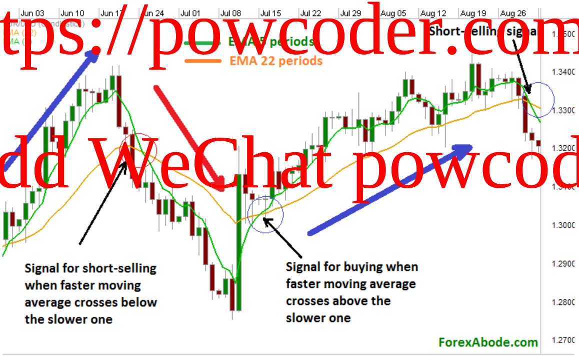

- `plot(x,y1,x,y2,x,y3..)` plots various times series on the same graph, `x,y1,y2,y3` are of the same length.
- `x` needs to be numerical and hence we use the date number `spy_t`
- `'b','g','r'` specify corresponding line color.
- `'--'` specifies line type: dashed line.
- `datetick('x')` converts the x ticks into date string.

Plot the MA time series



The MA Cross Trading Strategy

- The MA cross trading strategy uses technical analysis on past prices
  - **Buy signal** when the short-term trend cross-up long-term trend from bottom → buy and hold position when `sma_st>sma_lt`
  - **Sell signal** when short-term trend cross-down long-term trend from top → short sell position when `sma_st<sma_lt`



The MA Cross Trading Strategy


```
% buy when sma_st > sma_lt, sell when sma_st < sma_lt
position = (sma_st > sma_lt)*2 - 1;
```

- `sma_st > sma_lt` gives logical output 1 when it is true.
- `position=1*2-1=1` after a uplifting buy cross (`sma_st>sma_lt`)
- `position=0*2-1=-1` after a death sell cross (`sma_st<sma_lt`)
- Note: Code `position=(sma_st>sma_lt)` means a **passive** trading strategy that you buy the stock after a buy signal and sell the underlying after a sell signal: `position=0` when `sma_st<sma_lt`.
- Note: Code `position=(sma_st>sma_lt)*2-1` is the relative **active** trading strategy that you engage in short selling activity after a selling signal by setting `position =-1` when `sma_st<sma_lt`.

SubPlot


```
subplot(2,1,1)
plot(spy_t,spy_p, spy_t, sma_st, spy_t, sma_lt)
subplot(2,1,2)
plot(spy_t, position,'LineWidth',2)
```


- subplot(m,n,p) create a figure with various plots
  - The actual plot codes of plotting comes after subplot(m,n,p).
  - m,n define the plots layout structure: m rows n column panels.
  - p defines which sub-plot panel is defined in the following codes.

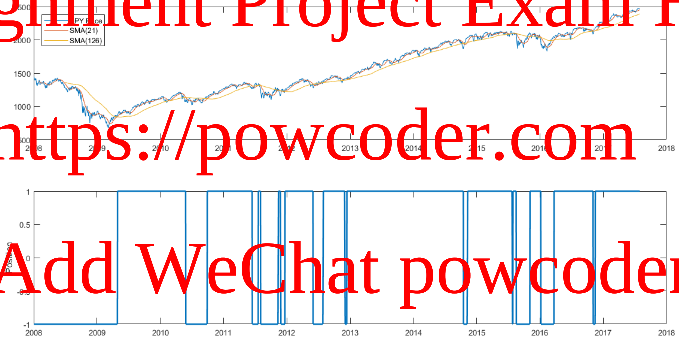
- In above code, we create a figure with 2x1 plot panels
  - the first panel plots the price and moving-average.
  - the second panel plots the position.

Plot the Trading Position


```
figure
subplot(2,1,1)
plot(spy_t, spy_p, spy_t, sma_st, spy_t, sma_lt)
legend('SPY Price', 'SMA(21)', ...
    'SMA(126)','Location','northwest')
dates_limits = [min(spy_t), max(spy_t)];
xlim(dates_limits)
datetick('x')

subplot(2,1,2)
plot(spy_t, position,'LineWidth',2)
xlim(dates_limits)
datetick('x')
```

Plot the Trading Position



Calculate Returns


```
%% Calculate returns
%daily and cumulative return on the MA strategy
strategy_ret =  position(2:end).*spy_lnret;
cumret_strategy = exp(cumsum(strategy_ret));
% cumulative return on market benchmark
cumret_market = exp(cumsum(spy_lnret));
```


- `strategy_ret` is the daily return based on the trading strategy
  - `(2:end)` as the 1st-day has no return yet;
  - note: we use element-by-element multiplication with the dot (.).

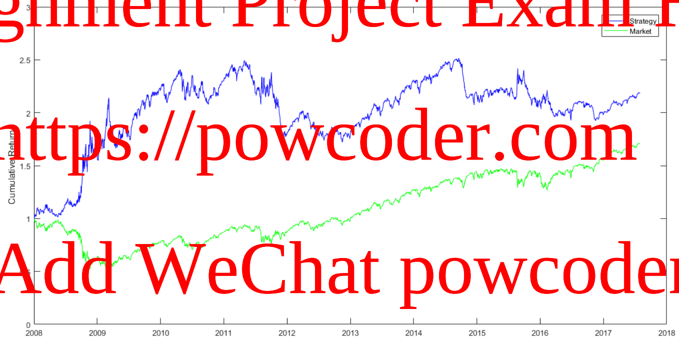
- Use `cumsum` as we work with ln return, and use the `exp` to convert back to cumulative simple return for comparison.
- The market benchmark assumes a buy and hold trading strategy.

Plot the Cumulative Return


```
figure
plot(spy_t(2:end), cumret_strategy,'b');
hold on % hold on the previous plot
plot(spy_t(2:end), cumret_market,'g');
legend('Strategy','Market')
dates_limits = [min(spy_t), max(spy_t)];
xlim(dates_limits)
datetick('x')
ylabel('Cumulative Return ')
```

- `hold on` retains plots in the current axes so that new plots added to the axes; otherwise, the new plots will overwrite the existing ones.
- `spy_t(2:end)` as we plot returns now.
- From the plot, it is clear that the trading strategy outperforms the market itself.

Plot the Cumulative Return



Annual Summary Statistics


```
%% Calculate Annual Return
annual_ret = mean(spy_lnret) * 252;
annual_std = std(spy_lnret) * sqrt(252);
% assuming the rf = 0.03
sharpe_ratio = (annual_ret - 0.03) / annual_std;

% combine results together to display
res = [annual_ret,annual_std,sharpe_ratio];
disp(['The annual return, stdeve and Sharpe Ratio are: ...
    ',num2str(res)])
```


- Annualise daily ln return by multiplying 252, the number of trading days in a year.
- Annualise standard deviation by multiplying square root of 252.
- Assume a relevant annual risk-free rate to calculate Sharpe ratio.

Construct Drawdown Curve

- A drawdown curve evaluates the performance of a trading strategy against its own best performance. $DD_t = \frac{HWM - R_t}{R_t}$
- At $t = 0$, set a high water mark (HWM) $= 1$.
- At $t = 1$, the trading strategy makes a return $= 1.5$
  - $1.5 >$ HWM $=1 \rightarrow$ new HWM $=1.5$, and drawdown $= 0$

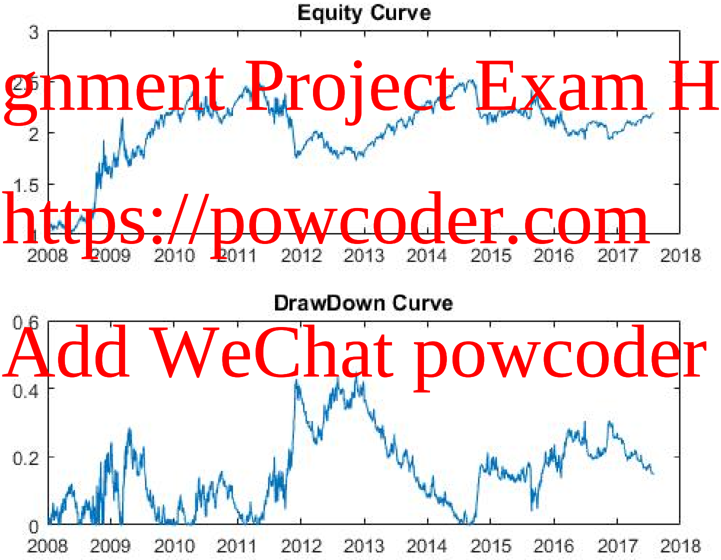
- At $t = 2$, the trading strategy makes a return, but lower $= 1.2$
  - $1.2 <$ HWM $=1.5 \rightarrow$ HWM $=1.5$ stays the same
  - drawdown $= (1.5-1.2)/1.2 = 0.25$

- A peak in the drawdown curve is the maximum loss in the profits from the highest profit point in history

Construct Drawdown Curve


```
%% Calculate the Drawdown Curve
% set HMW = 1 for start with.
high_water_mark = 1;
T = length(cumret_strategy);
dd_curve = zeros(T,1);
for t = 1:T
    if cumret_strategy(t) > high_water_mark
        high_water_mark = cumret_strategy(t);
    end
    drawdown = (high_water_mark - ...
        cumret_strategy(t))/cumret_strategy(t);
    dd_curve(t) = drawdown;
end
```

Plot the Drawdown Curve



Take Away

- Extract data from external resources and how to clear data.
- Calculating moving average and construct trading position.
- Calculate returns, cumulative returns and annualise relevant summary statistics
- The concept of drawdown curve in asset manager performance evaluation