WORLDQUANT UNIVERSITY

Econometrics Module 7

MSc Financial Engineering

Table of Contents

1. Brief

This document contains the core content for Module 7 of Econometrics, entitled Algorithmic Trading. It consists of eight sets of notes and one video script.

2. Course Context

Econometrics is the second course presented in the WorldQuant University (WQU) Master of Science in Financial Engineering (MScFE) program. In this course, you will apply statistical techniques to the analysis of econometric data. The course starts with an introduction to the R statistical programming languages that you will use to build econometric models, including multiple linear regression models, time series models, and stochastic volatility models. You will learn to develop programs using the R language, solve statistical problems, and understand value distributions in modeling extreme portfolio and basic algorithmic trading strategies. The course concludes with a review on applied econometrics in finance and algorithmic trading.

2.1 Course-level Learning Outcomes

Upon completion of the Econometrics course, you will be able to:

1. Write programs using the R language.
2. Use R packages to solve common statistical problems.
3. Formulate a generalized linear model and fit the model to data.
4. Use graphic techniques to visualize multidimensional data.
5. Apply multivariate statistical techniques (PCA, factor analysis, etc.) to analyze multidimensional data.
6. Fit a time series model to data.
7. Fit discrete-time volatility models.
8. Understand and apply filtering techniques to volatility modeling.
9. Understand the use of extreme value distributions in modeling extreme portfolio returns.
10. Define some common risk measures like VaR and Expected Shortfall.
11. Define and use copulas in risk management.
12. Implement basic algorithmic trading strategies.

2.2 Module Breakdown

The Econometrics course consists of the following one-week modules:

1. Basic Statistics
2. Linear Models
3. Univariate Time Series Models
4. Univariate Volatility Modeling
5. Multivariate Time Series Analysis
6. Introduction to Risk Management
7. Algorithmic Trading

3. Module 7:

Algorithmic Trading

In Module 7, you will be introduced to basic algorithmic trading strategies, and will learn some of the core concepts of algorithmic trading.

3.1 Module-level Learning Outcomes

After completing this module, you will be able to:

1. Understand the core concepts of algorithmic trading.
2. Understand different trading strategies.
3. Use nonlinear models in trading.
4. Understand Heath-Jarrow-Morton (HJM) framework.

3.2 Transcripts and Notes

3.2.1 Notes: Introduction

Algorithmic trading is defined as the buying and selling of financial instruments using predefined rules called algorithms. The algorithms are tested on historical data before being used in live trading. The emergence of this new field was mainly due to:

1. Computers
2. Quantitative algorithms
3. Research and Development in finance
4. Market liquidity
5. Lower transaction costs
6. The need to eliminate irrational trading
7. Lower regulation in certain economies

Nowadays there are different algorithmic trading strategies depending on the finance theory used, markets, quantitative methods as well as speed of transaction. Strategies can focus on:

1. Speed of transaction
2. Fundamental analysis
3. Technical analysis
4. Text mining

Quantitative methods used by traders are from:

1. Statistics
2. Machine Learning
3. Econometrics

4 Mathematics

Fundamental data used in quant trading includes:

1 Liquidity
2 Return on Assets (ROA)
3 Return on Equity (ROE)
4 Debt on equity ratio
5 Cash flow
6 Market capitalization

3.2.2 Notes: Algorithmic Trading Strategies

This set of notes will provide useful pointers about various types of algorithmic trading strategies, including mean-reversion strategies, various kinds of arbitrage and other techniques you may wish to adopt as a financial engineer. We will explore several strategies in more detail over the course of this module.

Momentum strategies

- Provide good results in rising markets.
- Focus on stocks that have a certain trend on a high volume.
- For momentum strategies, price differences are calculated at fixed time intervals.

Mean-reversion strategies

- Assets are expected to eventually return to their mean levels or equilibrium levels.
- If market price < average value, the asset is considered attractive as it is expected to increase (Buy signal).
- If market price > average value (Sell signal).

News-based trading

- This form of trading relies on algorithms to identify key words from Twitter, Facebook, Bloomberg, blogs and different websites in order to obtain trading signals.
- There is a recent boom in this type of trading due to the development of machine learning algorithms for text analysis and the huge amount of free data on the internet.

- Some companies sell data for market analysis (Sentdex).

Statistical arbitrage

- The algorithms identify arbitrage opportunities (small price discrepancies of the same asset on different markets).

Event arbitrage

- Certain events, such as company mergers or takeovers, or even company restructuring, are considered to generate certain asset price/return patterns.

Pairs trading

- It is difficult to find a tradable asset that has mean-reverting behavior.
- Equities behave like Geometric Brownian Motion (GBM) and make the mean-reverting trade strategies relatively useless.
- Pairs trading is the simplest form of mean reverting strategy.
- Pairs trading involves trading the stocks from two companies in the same that sector (for example Pepsi and Coca Cola stock; Renault and Peugeot-Citroen group). Stock prices are influenced mainly by the same shocks. Their relative stock prices will diverge due to certain events but will revert to the long-running mean.

Pure quant trading and Excel

There is a recent trend to look down on Excel as it is not considered a pure quantitative tool. Is Excel good for algorithm trading? Well, it depends. If the algorithms require sophisticated calculations and high speed of transaction, then Excel may not be the right tool. Excel can be an excellent tool for basic algorithm trading strategies and

lower frequency transactions. It can also be used to check how different basic algorithms perform. It is also good when you want to have access to data instantly.

3.2.3 Notes: Pair Trading

Trading correlated stocks (pairs trade or pair trading) means trading a pair of stocks that are highly correlated. This means the two stocks move in the same direction in most of the situations. The trading opportunity appears when the two stocks move temporarily in opposite directions (the correlation between them weakens for a period). This means that one stock will go up and the other goes down.

Pair trading strategy: Short the outperforming stock and long the underperforming stock.

Pairs trade is usually done with stocks issued by companies from the same sector. Possible pair trading opportunities are:

- Coca-Cola (KO) and Pepsi (PEP)
- Renault (RNL) and PSA Peugeot Citroen (UG)
- Exxon Mobil (XOM) and Chevron Corporation (CVX)

The strategy requires market timing and decision taking skills. However, it does not have much downside risk, and it can be implemented as an algorithmic trading strategy.

Stock price evolution is very difficult to be forecasted but the spread between two correlated stocks is easier to be predicted. The spread can be a stationary time series, and therefore a trader can use an Autoregressive Moving Average (ARMA) model to predict the spread. According to Ernest Chan, “a time series is stationary if it never drifts farther and farther away from its initial value”.

The spread can also be traded separately.
There may be a cointegration relation between the two stocks, considering the high correlation coefficient. Cointegration means that the two stock prices are not

stationary (as in many cases) but a linear combination between them is stationary. Ernest Chan calculated that there is less than 90% probability that Coca Cola and Pepsi stocks are cointegrated, despite their high correlation.

If two stocks are highly correlated that does not mean that they will remain highly correlated in the long term. Their correlation may weaken in the long term. The correlation coefficient between Coca Cola and Pepsi stocks increased significantly during the years. Ernest Chan calculated a correlation coefficient of 0.4849 between the two stocks, while our recent calculations indicated a correlation coefficient of 0.8421 for the first 11 months of 2015.

Example: Pair trading in Excel using basic statistics

We want to do pair trading using Coca-Cola (KO) and Pepsi (PEP) stocks. We download the stocks' closing prices from Yahoo Finance.

Frequency: daily

Period: January 2nd 2015 – November 26th 2015

The calculations will be done in Excel.

Pepsi stock closing price – cells B2:B229

Coca Cola stock closing price – cells C2:C229

The two stocks have to be tested if they are highly correlated before doing pair trading. The correlation coefficient will be calculated using CORREL function from Excel.

The correlation coefficient between Coca-Cola and Pepsi: CORREL (B2:B229,C2:C229)

The value returned by the function is 0.8421 which is close to 1. This means Coca-Cola and Pepsi stocks are highly correlated, so pair trading can be done in this case.

Calculate the difference between Pepsi stock price and Coca-Cola stock price (enter the following formula in D2 cell): B2-C2

The difference is then calculated for each day of the considered period.

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Pepsi	Coca Cola	Difference	Trading Signal						
2	1/2/2015	92.41	41.10	51.31		Correlation coefficient between the two stocks:				0.842119	
3	1/5/2015	91.71	41.10	50.62							
4	1/6/2015	91.02	41.41	49.61							
5	1/7/2015	93.68	41.92	51.76							
6	1/8/2015	95.38	42.43	52.95							
7	1/9/2015	94.74	41.96	52.77							
8	1/12/2015	94.35	41.58	52.76							
9	1/13/2015	94.28	41.57	52.70							
10	1/14/2015	94.59	41.50	53.09							
11	1/15/2015	94.59	41.33	53.26							
12	1/16/2015	95.20	41.48	53.72							
13	1/20/2015	95.41	42.09	53.32							
14	1/21/2015	95.30	42.28	53.02							

Suppose we want to do pair trading in November. The following algorithm is considered:

If the absolute value (difference at t-1 period – average value of difference)> 2* standard deviation of the difference

We trade as the gap between the two stock prices is widening

Else: we do not trade

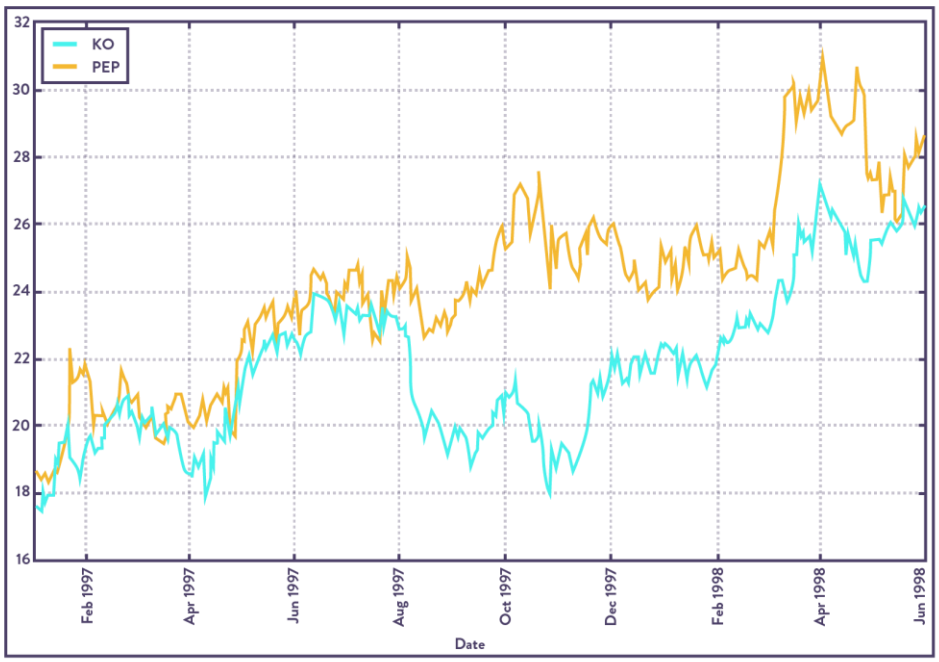
Formula introduced in D212 cell:

IF(ABS(D211-AVERAGE($D$2:D212))>2*STDEV.P($D$2:D212), "TRADE", "NOT TRADE")

212	11/2/2015	101.40	42.24	59.16	TRADE
213	11/3/2015	100.92	42.16	58.76	TRADE
214	11/4/2015	100.62	41.97	58.65	TRADE
215	11/5/2015	100.61	42.33	58.28	TRADE
216	11/6/2015	99.72	41.96	57.76	NOT TRADE
217	11/9/2015	98.88	41.54	57.34	NOT TRADE
218	11/10/2015	98.83	41.77	57.06	NOT TRADE
219	11/11/2015	99.43	42.04	57.39	NOT TRADE
220	11/12/2015	98.37	41.58	56.79	NOT TRADE
221	11/13/2015	98.04	41.38	56.66	NOT TRADE
222	11/16/2015	99.56	41.96	57.60	NOT TRADE
223	11/17/2015	98.83	41.67	57.16	NOT TRADE
224	11/18/2015	100.27	42.28	57.99	NOT TRADE
225	11/19/2015	100.93	43.11	57.82	NOT TRADE
226	11/20/2015	100.10	42.43	57.67	NOT TRADE
227	11/23/2015	100.85	42.96	57.89	NOT TRADE
228	11/24/2015	100.63	43.36	57.27	NOT TRADE
229	11/25/2015	100.50	43.36	57.14	NOT TRADE

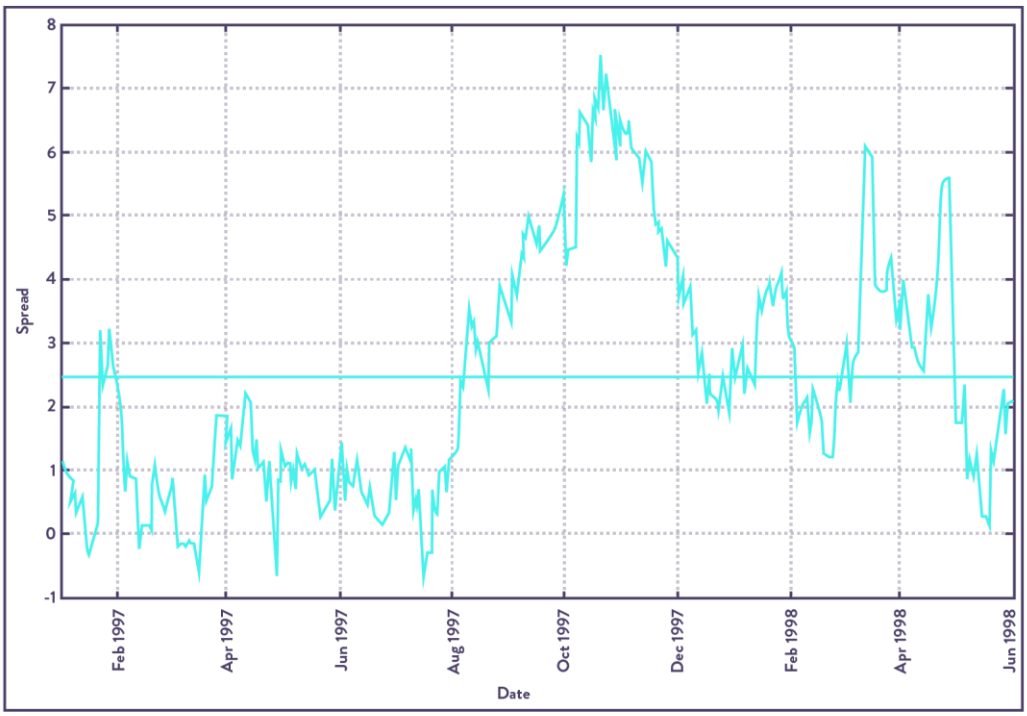
Example: Pair trading using econometrics

We use the same stocks. The period is different period and an econometric approach is adopted.



(Source: Michael Heydt – "Mastering Pandas for Finance")

- According to the chart, the two stocks share the same stochastic trend.
- The spread between the two stocks tends to be low in the first part of 1997. The spread widens in the second part.
- The visual inspection of the chart may not provide all the necessary information for trading, so spread has to be calculated.



(Source: Michael Heydt – "Mastering pandas for Finance")

- The econometric algorithm will provide z-score.
- The following simple regression will be implemented:

PEP = slope*KO + intercept

Spread = PEP – slope*KO – intercept

Spread can be seen as the residual value

z-score is calculated

z-score = (spread – 100-day rolling average of spread)/(100-day rolling standard deviation of spread)

So spread data is normalized on a 100-day rolling average.

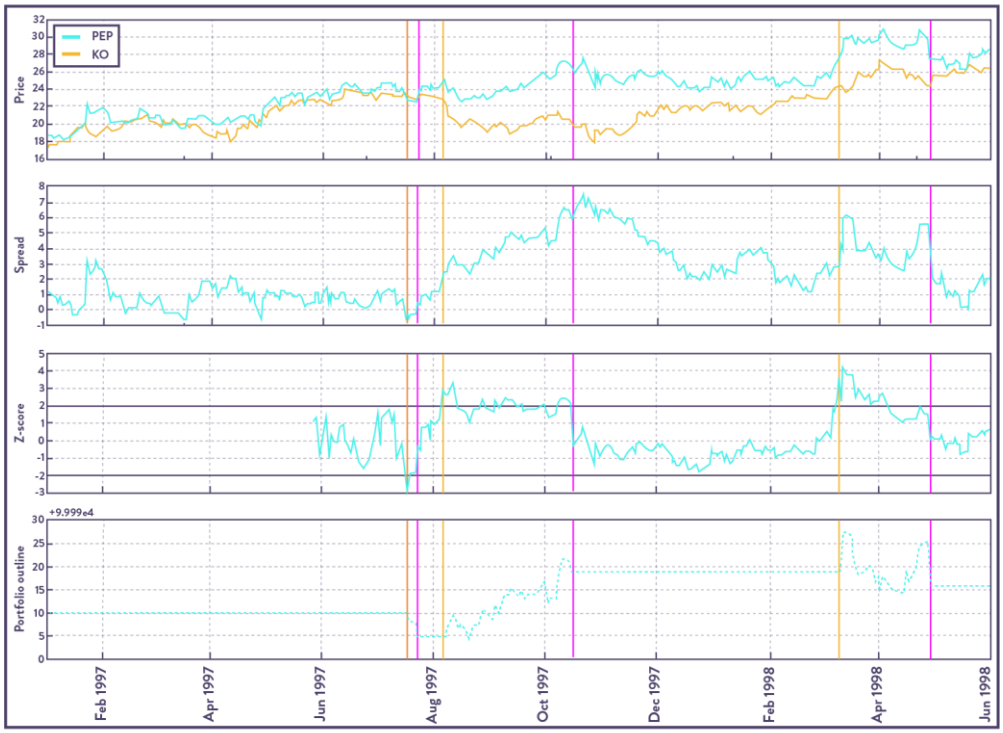
Trading strategy

If z-score > 2 buy (long) KO and sell (short) PEP.

If z-score < -2 buy PEP and sell KO.

If z-score < 0.5 sell the two stocks (limit our exposure).

The algorithm provided the following results:



(Source: Michael Heydt – "Mastering pandas for Finance")

- There are no trading opportunities if the spread is low and remains nearly constant (portfolio value is constant from January 1997 until August 1997).
- If the spread increases, trading opportunities appear (portfolio value increased as z-score was above 2).
- If the spread drops, some losses appeared (the first part of 1998).

This strategy is presented in more detail in Michael Heydt's book, *Mastering Pandas for Finance* (2015).

3.2.4 Notes: Trading Using Google Trends

The article "Quantifying Trading Behavior in Financial Markets Using Google Trends" showed that we can trade efficiently if we take into consideration the frequency of certain key words in Google searches. Tobias Preis, Helen Susannah Moat, and H. Eugene Stanley, the authors of the article, propose the following strategy:

1 Collect data about Dow Jones Industrial Average (DJIA)

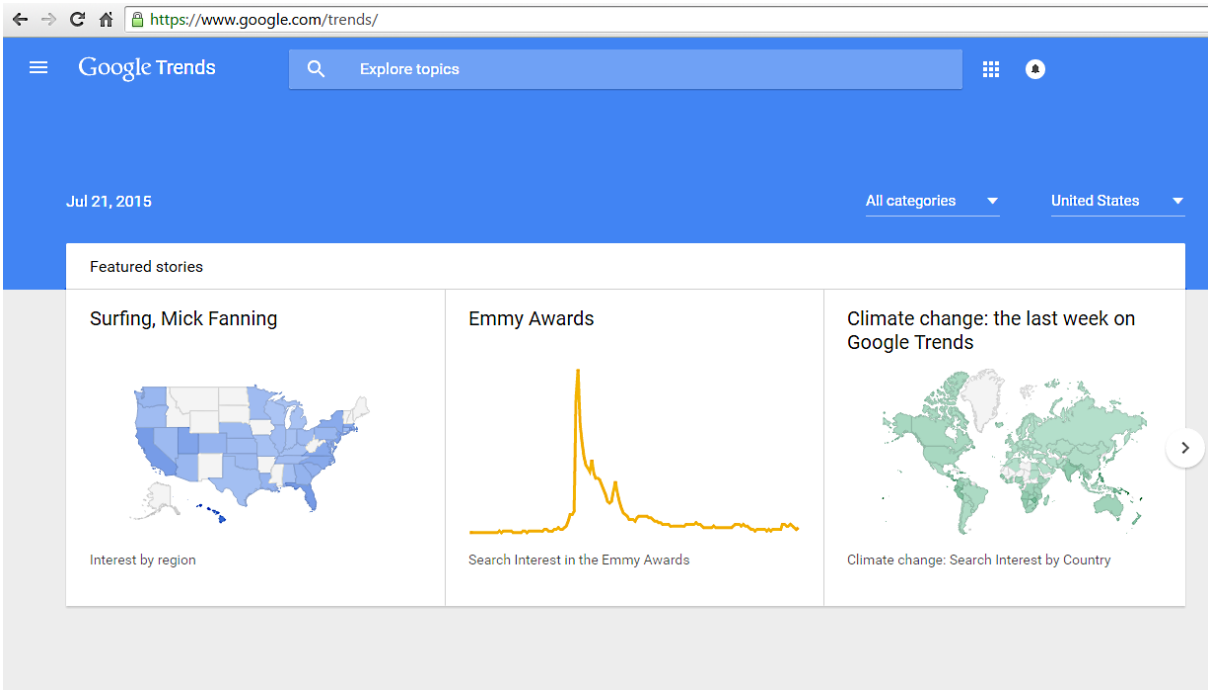
DJIA was downloaded from S&P Dow Jones Indices LLC:

Period: 2005-07-18 – 2015-07-16

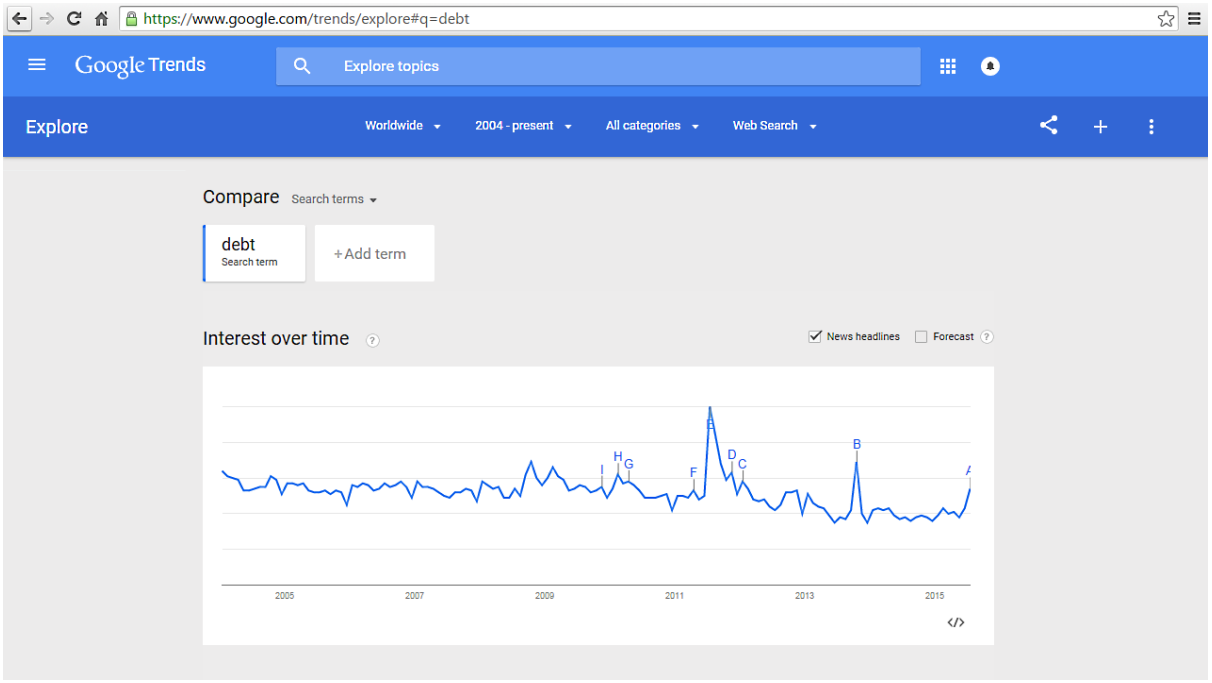
Frequency: weekly

39	DATE	DJIA
40	2005-07-23	10637.93
41	2005-07-30	10631.96
42	2005-08-06	10634.52
43	2005-08-13	10606.64
44	2005-08-20	10562.54
45	2005-08-27	10474.45
46	2005-09-03	10452.89
47	2005-09-10	10624.31
48	2005-09-17	10605.19
49	2005-09-24	10451.76
50	2005-10-01	10498.88
51	2005-10-08	10374.67
52	2005-10-15	10242.55

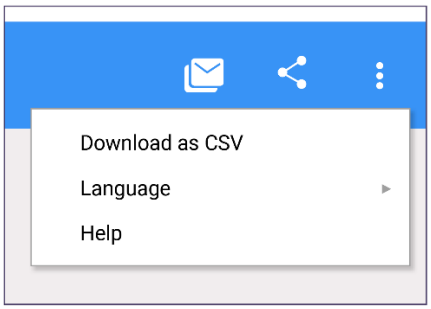
2 Collect Google trends data



The authors propose to search the key word "debt" in Google trends:



3 Download data in CSV format



	A	B	C
1	Web Search interest: debt		
2	Worldwide; 2004 - present		
3			
4	Interest over time		
5	Week	debt	
6	2004-01-04 - 2004-01-10	36	
7	2004-01-11 - 2004-01-17	36	
8	2004-01-18 - 2004-01-24	37	
9	2004-01-25 - 2004-01-31	37	
10	2004-02-01 - 2004-02-07	37	
11	2004-02-08 - 2004-02-14	37	
12	2004-02-15 - 2004-02-21	36	
13	2004-02-22 - 2004-02-28	35	
14	2004-02-29 - 2004-03-06	29	
15	2004-03-07 - 2004-03-13	35	
16	2004-03-14 - 2004-03-20	34	

Unpacking the trading strategy

Searches for a Google term have increased at the end of a three-week period -> expected upturn in the market -> long position. Searches for a Google term have decreased at the end of a three-week period -> expected downturn in the market -> short position.

In this case the trading strategy is based on the word debt.

The authors' strategy is complex, as they use 98 different words in their Google trend searches.

$p(t)$ – DJIA closing price

$n(t-1)$ – number of searches have been carried out in Google Trends in week t-1

$$\Delta n(t, \Delta t) = n(t) - N(t-1, \Delta t),$$

$$N(t-1, \Delta t) = (n(t-1) + n(t-2) + \ldots + n(t-\Delta t))/\Delta t,$$

$\Delta t = 3$ (number of weeks).

	A	B	C	D	E	F	G
1	Web Search interest: debt						
2	Worldwide; 2004 - present						
3							
4	Interest over time						
5	Week	debt	$\Delta n(t,\Delta t)$	Order signals			
6	2004-01-04 - 2004-01-10	36	-	-			
7	2004-01-11 - 2004-01-17	36	-	-			
8	2004-01-18 - 2004-01-24	37	-	-			
9	2004-01-25 - 2004-01-31	37	0.67	1	if	$\Delta n(t-1,\Delta t) > 0$	
10	2004-02-01 - 2004-02-07	37	0.33	1	if	$\Delta n(t-1,\Delta t) < 0$	
11	2004-02-08 - 2004-02-14	37	0.00	-1			
12	2004-02-15 - 2004-02-21	36	-1.00	-1			
13	2004-02-22 - 2004-02-28	35	-1.67	-1			
14	2004-02-29 - 2004-03-06	29	-7.00	-1			

C9=B9-AVERAGE(B6:B8)

D9=IF(C9>0,1,-1)

Further reading

Preis, T., Moat, H. and Stanley, H. (2013). Quantifying Trading Behavior in Financial Markets Using Google Trends. *Scientific Reports*, 3(1). Available at ResearchGate.

3.2.5 Notes: Algorithmic Trading in R

Extracting Dow Jones Index from Yahoo Finance

Technical indicators are implemented in the `quantmod` package. The `quantmod` package loads `xts` and `zoo` packages.

Downloading quantmod package in R

R code:

```
library("quantmod")
```

```
Loading required package: xts
Loading required package: zoo
```

The next step is to extract Dow Jones Index (DJI) data from Yahoo Finance.

DJI data contains the following columns:

1. DJI.Open
2. DJI.High
3. DJI.Low
4. DJI.Close

We select close value.

R code:

```
getSymbols("^DJI",src="yahoo")
dow_jones<- DJI[,"DJI.Close"]
dow_jones
```

```
                DJI.Close
2007-01-03        12474.52
2007-01-04        12480.69
2007-01-05        12398.01
2007-01-08        12423.49
2007-01-09        12416.60
2007-01-10        12442.16
2007-01-11        12514.98
2007-01-12        12556.08
2007-01-16        12582.59
```

Extracting DJI data between two specified dates

R code:

```
dow_jones<- dow_jones[(index(dow_jones) >= "2010-01-01" &
index(dow_jones) <= "2017-09-10"),]
dow_jones
```

Calculating daily returns using closing prices

R code:

```
return_dow_jones<-Delt(dow_jones, k=1)
return_dow_jones
```

```
           Delt.1.arithmetic
2010-01-04                NA
2010-01-05     -1.128163e-03
2010-01-06      1.570331e-04
2010-01-07      3.138043e-03
2010-01-08      1.068184e-03
2010-01-11      4.313334e-03
2010-01-12     -3.444346e-03
```

Calculating returns for different periods can be done by changing the k parameter. For example, if we want to calculate returns for lag 1 to lag 3, we would use the k=1:3 command.

R code:

```
return_dow_jones2<-Delt(dow_jones, k=1:3)
return_dow_jones2
```

```
           Delt.1.arithmetic  Delt.2.arithmetic  Delt.3.arithmetic
2010-01-04                NA                 NA                 NA
2010-01-05     -1.128163e-03                 NA                 NA
2010-01-06      1.570331e-04      -9.713069e-04                 NA
2010-01-07      3.138043e-03       3.295569e-03       2.163688e-03
2010-01-08      1.068184e-03       4.209579e-03       4.367273e-03
2010-01-11      4.313334e-03       5.386126e-03       8.541071e-03
```

3.2.6 Notes: Momentum Trading and Backtesting

Momentum trading

Momentum trading (also known as **directional trading**) is when the financial instrument has an upward/downward trend that the financial engineer can use in his or her trading. For example, a long-term trend of the financial asset can provide successful trades in this case. Sometimes, there are situations when we cannot identify a clear trend in a financial instrument, for example, if the asset price fluctuates randomly.

Let's consider the DJI example. According to the graph bellow, DJI has experienced an upward trend in the past years, so a momentum trading strategy can be applied in this case.

Backtesting

Backtesting means simulating an investment strategy based on historical data. It is also used for calibrating and evaluating an investment strategy. Various indicators related to risk and return are calculated.

Backtesting the strategy requires splitting the data in two smaller datasets:

- In-sample dataset (consisting of 70–80% of the data)
- Out-sample dataset (20–30% of the data)

The in-sample data is used for parameter estimation and to evaluate the performance. The estimated parameters are used in the out-sample data to check their generalization capacity. If the tests using out-sample data provide similar results with in-sample test, then we validate the trading strategy and we can use it in live trading.

You will need to indicate the start and end dates for in-sample and out-sample periods.

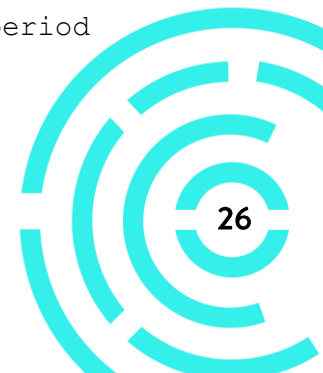
```
in_sd -> in-sample data starts
in_ed -> in-sample data ends

out_sd -> out-sample data starts
in_ed -> in-sample data ends
```

R code:

```
in_sd<- "2010-01-01"
in_ed<- "2015-12-31"
out_sd<- "2016-01-01"
out_ed<- "2017-09-10"


in_dow_jones -> the Dow Jones closing price for in-sample period
in_return_dow_jones -> the Dow Jones return for in-sample period
```



R code:

```
in_dow_jones<- dow_jones[(index(dow_jones) >= in_sd&
index(dow_jones) <= in_ed),]
in_dow_jones
in_return_dow_jones<- return_dow_jones[(index(return_dow_jones) >=
in_sd& index(return_dow_jones) <= in_ed),]
in_return_dow_jones

out_dow_jones -> the Dow Jones closing price for out-sample period
out_return_dow_jones -> the Dow-Jones return for out-sample period
```

Splitting the dataset into two categories (in-sample and out-sample) helps to control human bias towards parameter estimation.

R code:

```
out_dow_jones<- dow_jones[(index(dow_jones) >= out_sd&
index(dow_jones) <= out_ed),]
out_dow_jones
out_return_dow_jones<- return_dow_jones[(index(return_dow_jones)
>= out_sd& index(return_dow_jones) <= out_ed),]
out_return_dow_jones
```

The next step is to generate automated trading signals using:

1. Moving average convergence divergence (MACD)
2. Bollinger band indicators

R functions used:

```
MACD
BBands
```

Step 1: MACD parameters can be adjusted considering the trading strategy.

R code:

```
macd<- MACD(in_dow_jones, nFast =12, nSlow = 26, nSig = 9,
maType="SMA", percent= FALSE)
bollinger_band<- BBands(in_dow_jones, n = 20, maType="SMA", sd
= 2)
```

The trading strategy is based on the following outcomes:

- Signal is initialized with NULL
- If DJI > upper Bollinger band and MACD value > MACD signal -> Buy signal (1)
- If DJI < lower Bollinger band and MACD value < MACD signal -> Sell signal (-1)
- Out of the market -> Signal 0

R code:

```
Signal<-NULL
signal <- ifelse(in_dow_jones> bollinger_band[,'up']
&macd[,'macd']>macd[,'signal'],1,ifelse(in_dow_jones<
bollinger_band[,'dn'] &macd[,'macd']<macd[,'signal'],-1,0))
```

Note the following about this process:

- No transaction costs are included in the analysis.
- The mentioned strategy can be used for both long and short positions.
- A long only or short only strategy can also be implemented.
- The exit criterion can be changed.

Step 2: Trade return is calculated using the following R code:

```
trade_return<- in_return_dow_jones*lag(signal)
trade_return
```

Step 3: The strategy performance can be checked using the `PerformanceAnalytics` package.

- We install the package using the following R code:

```
install.packages("PerformanceAnalytics")
library(PerformanceAnalytics)
```

Step 4: The cumulative return is calculated using the following R code:

```
cumm_return<- Return.cumulative(trade_return)
cumm_return
```

```
                          Delt.1.arithmetic
Cumulative Return              0.01906121
```

Step 5: The annualized return is calculated as follows:

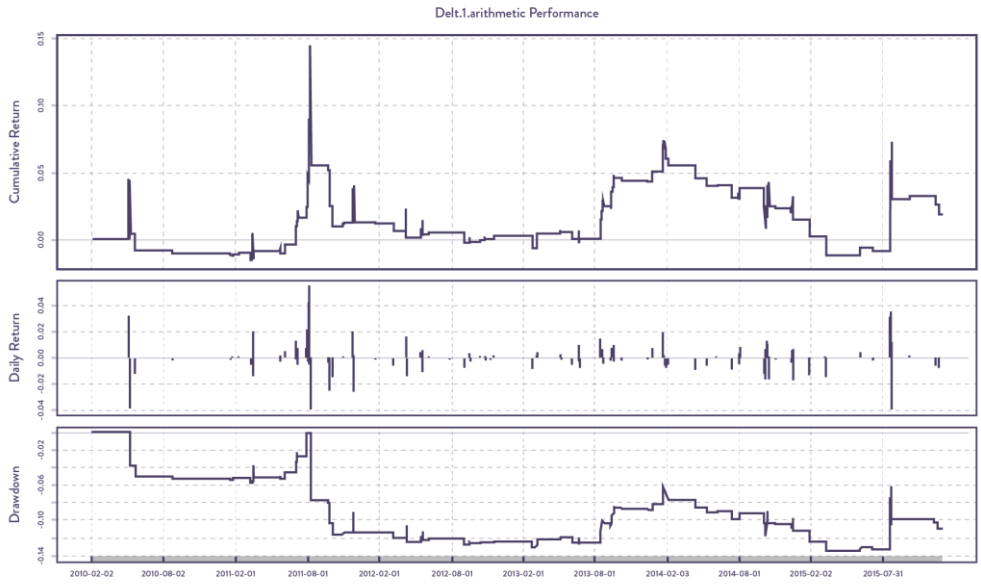
```
annual_return<- Return.annualized(trade_return)
annual_return
```

```
                          Delt.1.arithmetic
Annualized Return             0.003198539
```

Step 6: `Charts.PerformanceSummary` plots cumulative and daily returns along with drawdown at a given point of time.

R code:

```
charts.PerformanceSummary(trade_return)
```



Step 7: The details about the trade performance are provided by summary function, generated as follows in R code:

Comments regarding charts:

- Cumulative returns are positive at the end of in-sample period, which means the strategy is profitable. However there are periods when the cumulative returns become negative,
- The strategy should be seen with high caution as it has to be tested for out-sample period
- There are periods with high volatility and periods with low volatility according to daily returns chart and cumulative returns. This may indicate volatility clustering often encountered on financial markets data.

```
summary(as.ts(trade_return))

    Min.   1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
-0.03977  0.00000  0.00000  0.00002  0.00000  0.05546       20
```

- The maximum drawdown is calculated (it is 13.61 % in this case).

R code:

```
maxDrawdown(trade_return)
```

```
[1] 0.1360742
```

- Daily and annualized standard deviations are calculated for trade returns.

R code:

```
StdDev(trade_return)
```

```
                                  [,1]
                StdDev 0.003920434
```

```
StdDev.annualized(trade_return)
```

```
                                  Delt.1.arithmetic
Annualized Standard Deviation           0.06223497
```

- Value-at-Risk (VaR) is calculated

R code:

```
VaR(trade_return, p = 0.95)
```

- The Sharpe ratio on a daily and annualized basis is calculated.

R code:

```
SharpeRatio(as.ts(trade_return), Rf = 0, p = 0.95, FUN =
"StdDev")
```

```
                                                    [,1]
StdDev Sharpe (Rf=0%, p=95%): 0.005186737
```

```
SharpeRatio.annualized(trade_return, Rf = 0)
```

```
                                  Delt.1.arithmetic
Annualized Sharpe Ratio (Rf=0%)         0.05139457
```

Out-sample data analysis

The trading strategy above provides good results for in-sample data. The strategy is now tested for out-sample data.

Step 1: Moving average and Bollinger bands are calculated for out-sample data using the following R code:

```
macd<- MACD(out_dow_jones, nFast = 7, nSlow = 12, nSig =
15,maType="SMA", percent = FALSE)
bollinger_band <- BBands(out_dow_jones, n = 20, maType="SMA",
sd = 2)
```

Step 2: Signals are generated for out-sample data.

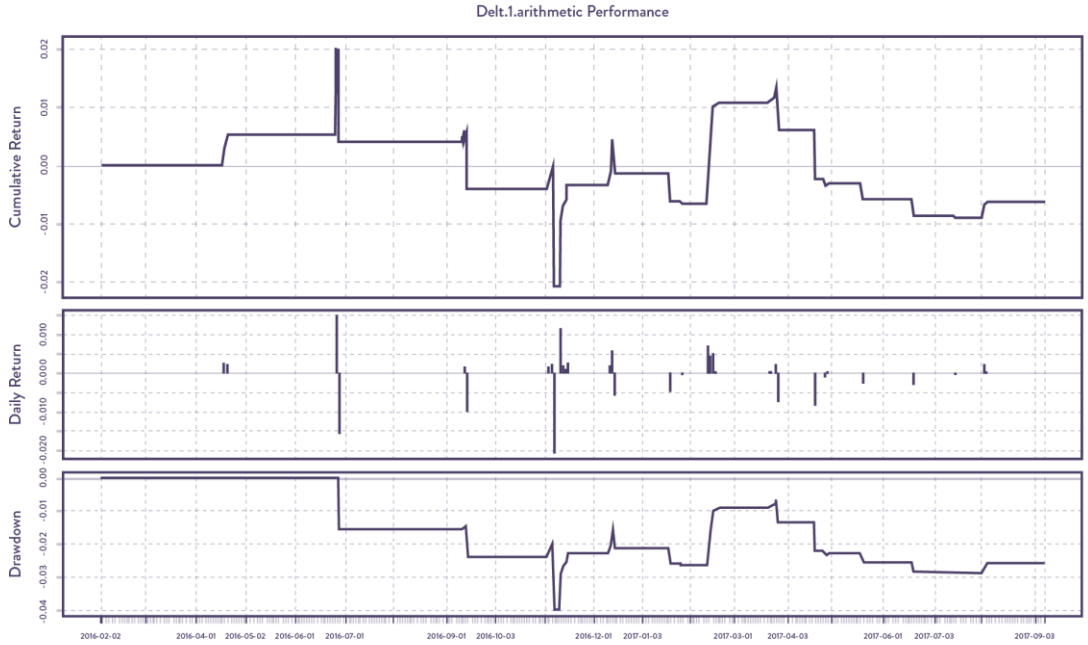
R code:

```
signal <- NULL
signal <- ifelse(out_dow_jones> bollinger_band[,'up']
&macd[,'macd']>macd[,'signal'],1,ifelse(out_dow_jones<
bollinger_band[,'dn'] &macd[,'macd']<macd[,'signal'],-1,0))
```

R code:

```
trade_return<- out_return_dow_jones*lag(signal)
```

```
cumm_ret<- Return.cumulative(trade_return)
annual_ret<- Return.annualized(trade_return)
charts.PerformanceSummary(trade_return)
maxdd<- maxDrawdown(trade_return)
sd<- StdDev(trade_return)
sda<- StdDev.annualized(trade_return)
VaR(trade_return, p = 0.95)
SharpeRatio(as.ts(trade_return), Rf = 0, p = 0.95, FUN =
"StdDev")
SharpeRatio.annualized(trade_return, Rf = 0)
```



Comments:

- The cumulative returns are positive in the first period, ending in negative ground towards the end. This means the strategy is not profitable. However, the strategy must be tested on different stocks and periods before reaching a final conclusion regarding profitability.
- The strategy is basic as it relies only on technical indicators. However, it can be improved by introducing trading signals generated by ARMA or GARCH forecasts.

- ARMA and GARCH models are generally used for short-term predictions (t+1), while VAR/VEC can be used for generating longer-term trends considering different macro or financial indicators.
- An updated version of the strategy can be initially tested in Excel, before being implemented in Python or R.

3.2.7 Notes: Modeling Stock Returns

Quantitative analysts have been trying for decades to find an efficient way of forecasting stock returns. This topic is presented in thousands of working papers, articles, books and websites.

Can we really predict stock returns? The answer is that sometimes we can, sometimes we cannot.

If stock returns follow a smooth process, we can forecast the evolution accurately. If returns tend to be a random process, then it is sometimes impossible to predict and we have to rely on our "feel of the data" (expert judgement). Certain models offer us some answers regarding modeling stock returns.

The Dividend Discount Model

DDM, also known as Gordon Growth Model, is a quantitative approach for valuing a stock price. According to DDM, the price of a stock is the sum of future dividends discounted back to their present value.

P – Current stock price

g – Constant growth rate in perpetuity expected for the dividends

r – Constant cost of equity capital for that company

D_1 – Next year's dividends

t – Discrete time

$$P = \frac{D_1}{r-g}.$$

Obtaining the equation

$$D_0(1+g)^t,$$

$$P = \sum_{i=1}^{\infty} D_0 \frac{(1+g)^t}{(1+r)^t},$$

$$s = \frac{(1+g)}{(1+r)},$$

$$P = D_0 s(1 + s + s^2 + \cdots).$$

$$\text{If } s < 1 \text{ then } 1 + s + s^2 + \cdots = \frac{1}{1-s},$$

P becomes:

$$P = D_0 \frac{1+g}{1+r} \frac{1}{1-\frac{1+g}{1+r}},$$

$$D_1 = D_0(1+g),$$

$$P = \frac{D_1}{1+r} \frac{1}{1-\frac{1+g}{1+r}} = \frac{D_1}{1+r} \frac{1+r}{1+r-1-g} = \frac{D_1}{r-g},$$

$$P = \frac{D_1}{r-g}.$$

If $g = 0$

$$P = \frac{D_1}{r}.$$

Stock price according to DDM > Stock price – The stock is undervalued.

Stock price according to DDM < Stock price – The stock is overvalued.

Disadvantages:

- It assumes a constant growth rate which is not true in most cases.
- Growth rate is lower when compared to the cost of capital.
- It assumes that the stock pays dividend.
- Stock price resulting from DDM is hyper-sensitive to the growth rate.

Autoregressive processes and random walks

We can forecast stock returns assuming that they follow an AR(1) process.

y – stock returns

$y_t = \alpha y_{t-1} + \varepsilon_t$ – AR(1) process

ε_t is normally distributed with mean 0 and variance $\sigma_\varepsilon^2 = 1$

ε_t and ε_s are independent for all $t \neq s$

α estimated from the historical data on y

- Random walk: $\alpha = 1$ – the current stock return provides no information about the future movement of the return.
- AR(1): $-1 < \alpha < 1$ – mean-reverting process. If the stock provides a high return currently, it is expected that the return will post a downward trend in the future.

Example:

Microsoft stock return – AR(1) process.

$\alpha = 0.75$ – Microsoft stock provides a daily return of 1 % currently. It is expected to provide a daily return of 0.75 % the next day.

$\alpha = 1$ – Microsoft stock return is a random process. Microsoft stock provides a daily return of 1 % currently. We cannot forecast the daily return for the next day.

VAR methodology

We can forecast stock returns using a vector autoregressive (VAR)[1] model.

$$x_{t+1} = Ax_t + \epsilon_{t+1}$$

x_t – vector of state variables including stock returns

$$E_t x_{t+1+j} = A^{j+1} x_t$$

[1] Note that this differs from Value-at-Risk, or VaR.

Variables that explain expected stock return:

- Lagged stock return
- Dividend yield
- P/E ratio
- Trading volume
- Default spread
- Yield on T-bill
- Change in yield of T-bill
- Term spread
- yield spread between overnight fixed income security and T-bill
- January dummy
- Growth rate of industrial production
- Change in inflation or unexpected inflation

Hurst exponent

The Hurst exponent is a simple indicator to calculate the predictability of an asset. It can take values between 0 and 1.

- Hurst exponent < 0.50 – asset price cannot be predicted using a quantitative model.
- Hurst exponent = 0.50 – asset price can be forecasted, but it is a very difficult task.
- Hurst exponent > 0.50 – asset price can be predicted using a quantitative technique, having a trend behavior.

Further reading

John Y. Campbell – *Present Value Relations and Stock Return Predictability*, Ec2723 Asset Pricing I Class Notes, Fall 2006

Dorothée Rouzet – *Discounted dividends and asset prices* Ec 1011b, Spring 2010

3.2.8 Notes: Artificial Neural Networks in Algorithmic Trading

The Artificial Neural Network (ANN) is used to model nonlinear processes that do not have a clear functional form. The model can be used to simulate chaos, which is frequently encountered in quantitative finance when modeling asset prices/returns.

Defining chaos

A sequence $\{y_t\}$ is said to be chaotic if it is generated from a deterministic difference equation and it does not explode / converge to a constant or repetitive cycle.

The Artificial Neural Network

$$y_t = a_0 + a_1 y_{t-1} + \sum_{i=1}^{n} a_i f_i(y_{t-1}) + \varepsilon_t \quad (1)$$

where $f_i(y_{t-1})$ is the cumulative distribution of a logistic function and a_1 is a constant.

In the case of a logistic function:

$$y_t = a_0 + a_1 y_{t-1} + \sum_{i=1}^{n} \alpha_i \left[1 + \exp\left(-\gamma_i (y_{t-1} - c_i)\right)\right]^{-1} + \varepsilon_t \quad (2)$$

- ANN is similar to the logistic-smooth transition autoregressive (LSTAR) model. The main difference is that only the intercept is time-varying when using ANN models.
- ANN can also be seen as an extension of the AR(1) model.
- ANN uses different logistic functions (nodes).
- For a large value of n, ANNs approximate any first order nonlinear model randomly chosen according to Kuan and White (1994)

ANN advantages:

- Models nonlinear processes which can be found in a diverse range of activities – e.g. finance, weather forecasting, pattern recognition, etc.
- Fits the data very well.
- Has adaptive elements.

ANN disadvantages:

- Does not have a clear economic interpretation.
- Overfitting of the data may occur, considering the large number of parameters to be estimated.

Optimization in ANN

Numerical optimization in ANN is complicated as there are many l. local minima. Therefore, it is hard to find the parameter that minimizes the sum of squared residuals. White (1989) indicates a method for surpassing this problem by means of recursive learning.

The algorithm is the following:

Obtain the *nonlinear least squares estimates* of the parameters using the first observations of the data where:

$\hat{\theta}_t$ is the vector of estimated parameters using these observations,

$\hat{y}_{t+1}$ is the predicted value of y_{t+1}, and

$\hat{\theta}_t$ acts as an initial condition in the difference equation.

The value of $\hat{\theta}_t$ is an initial condition in the difference equation:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \eta_t(y_{t+1} - \hat{y}_{t+1}). \quad (3)$$

η_t is the multiple vector of partial derivatives of equation (1) with respect to the parameters evaluated at the point estimates of θ_t .

The successive values of $\hat{\theta}_{t+1}$ are obtained until all the parameter estimates converge. White's approach is an efficient way of simulating chaos.

Algorithmic Trading with Neural Networks

The following exercise will allow you to practice skills and concepts learned in this module.

Overview

Market volatility can be low and it is not indicated to buy or sell the asset if there is not a clear trend. Therefore, we need a model that indicates three types of trend.

The solution is the Artificial Neural Network (ANN).

Input data -> ANN -> Output data

Input data -> DJI closing price

Output data -> Three classes (Up, Down or Nowhere).

Extracting DJI data from Yahoo Finance	
Loading `quantmod` package	`library("quantmod")`
Loading DJI data from Yahoo Finance	`getSymbols("^DJI",src="yahoo")`
Extracting DJI from Yahoo Finance	`dow_jones<- DJI[,"DJI.Close"]` `dow_jones`
Calculating technical indicators	
Implementing return	`return <- Delt(dow_jones)`

Implementing moving average	`average10<- rollapply(dow_jones,10,mean)`
	`average10`
	`average20<- rollapply(dow_jones,20,mean)`
	`average20`

Algorithmic Trading with Neural Networks	

Implementing standard deviation	`std10<- rollapply(dow_jones,10,sd)`
	
	`std20<- rollapply(dow_jones,20,sd)`

Implement RSI	`rsi5<- RSI(dow_jones,5,"SMA")`
	
	`rsi14<- RSI(dow_jones,14,"SMA")`

Implement MACD	`macd12269<- MACD(dow_jones,12,26,9,"SMA")`
	
	`macd7205<- MACD(dow_jones,7,20,5,"SMA")`

Implement bollinger bands	`bollinger_bands<-` `BBands(dow_jones,20,"SMA",2)`
Generate directions Return over the last 20 days > 2 % -> *Up* direction Return over the last 20 days is between -2 % and 2 % -> *Nowhere* Return over the last 20 days < - 2 % -> *Down* direction	
Generate a data frame named direction which consists of NA and a number of rows the same as the number of rows in dow_jones and one column	`direction<-` `data.frame(matrix(NA,dim(dow_jones)[1],1))`
Calculate the return over the last 20 days (20 is not a fixed value in quantitative finance. You can chose any value you consider)	`lagreturn<- (dow_jones - Lag(dow_jones,20)) /` `Lag(dow_jones,20)`

Indicate Up, Down and Nowhere directions	`direction[lagreturn> 0.02] <- "Up"`
	
	`direction[lagreturn< -0.02] <- "Down"`
	
	`direction[lagreturn< 0.02 &lagreturn> -0.02]`
	`<- "NoWhere"`

Creating `dow_jones` dataset using `cbind` function	

Closing price and indicators are put into one variable called dow_jones	`dow_jones<- cbind(dow_jones,average10,average20,std10,std20,rsi5,rsi14,macd12269,macd7205,bollinger_bands)`
Dividing neural network dataset in three parts: • Training dataset -> training the neural network • Validating dataset -> validating the estimated parameters • Testing dataset -> measure the accuracy of the prediction	

Indicate end and start dates for train, validating and testing period	`train_sdate<- "2010-01-01"`
	`train_edate<- "2013-12-31"`
	`vali_sdate<- "2016-01-01"`
	`vali_edate<- "2016-12-31"`
	`test_sdate<- "2017-01-01"`
	`test_edate<- "2017-09-10"`

Constructing date ranges for the three datasets	

Generate row numbers where the date is greater than and equal to start date and less than equal to the end date. Which() function is used.	`trainrow<- which(index(dow_jones) >= train_sdate& index(dow_jones) <= train_edate)`
	`valirow<- which(index(dow_jones) >= vali_sdate& index(dow_jones) <= vali_edate)`
	`testrow<- which(index(dow_jones) >= test_sdate& index(dow_jones) <= test_edate)`

Extract data for training, validating and testing periods	

Extracting data for training, validating and testing periods	`traindji<- dow_jones[trainrow,]`
	`validji<- dow_jones[valirow,]`
	`testdji<- dow_jones[testrow,]`

Calculate mean and standard deviation of the training data	`trainme<- apply(traindji,2,mean)` `trainstd<- apply(traindji,2,sd)`
Create three matrices of dimensions equal to the	`trainidn<- (matrix(1,dim(traindji)[1],dim(traindji)[2]))`

Training, validating and testing data dimensions	`valiidn<-`
	`(matrix(1,dim(validji)[1],dim(validji)[2]))`
	
	`testidn<-`
	`(matrix(1,dim(testdji)[1],dim(testdji)[2]))`

Normalize the three datasets.
T() function is used for matrix transposing.
	`norm_traindji<- (traindji -`
	`t(trainme*t(trainidn)))`
	`/t(trainstd*t(trainidn))`
	
	`norm_validji<- (validji -`
	`t(trainme*t(valiidn))) /`
	`t(trainstd*t(valiidn))`
	
	`norm_testdji<- (testdji -`
	`t(trainme*t(testidn))) /`
	`t(trainstd*t(testidn))`

Define training, validating and testing period	`traindir<- direction[trainrow,1]`
	
	`validir<- direction[valirow,1]`
	
	`testdir<- direction[testrow,1]`

Install ANN package (nnet)	
	`install.packages("nnet")`
	`library(nnet)`
Implement ANN	
The neural network starts with random weights and will provide different results. Seed() function will allow the same output every time	`set.seed(1)`

Implement ANN
Parameter 1 -> All normalized columns
Parameter 2 -> Target vector
Parameter 3 -> number of hidden layers (4 in this case)
	`neural_network<-`
	`nnet(norm_traindji,class.ind(traindir),size=4`
	`,trace=T)`
	
	`neural_network`

Parameter 4 -> Output is indicated at the end (trace=T) or not (trace=F).	
Obtain data dimension	`dim(norm_traindji)`

Make predictions	`vali_pred<-`
	`predict(neural_network,norm_validji)`
	
	`head(vali_pred)`

Calculate the predicted direction using the information obtained above.
Threshold = 0.50
Value > 0.50 -> obtain directions
The first line -> creates a data frame of length equal to the vali_pred length.
The next lines are used for checking the condition.
	`vali_pred_class<-`
	`data.frame(matrix(NA,dim(vali_pred)[1],1))`
	
	`vali_pred_class[vali_pred[,"Down"] > 0.5,1]`
	`<- "Down"`
	
	`vali_pred_class[vali_pred[,"NoWhere"] >`
	`0.5,1] <- "NoWhere"`
	
	`vali_pred_class[vali_pred[,"Up"] > 0.5,1] <-`
	`"Up"`

Check forecasts accuracy	

Load caret library
Use confusionMatrix() over the predicted class and original class for the validating dataset.
	`library(caret)`
	
	`matrix<-`
	`confusionMatrix(vali_pred_class[,1],validir)`
	
	`matrix`

Check the accuracy on testing data	`test_pred<-`
	`predict(neural_network,norm_testdji)`
	
	`test_pred`

Indicate the classes for the testing data	`test_pred_class<-`
	`data.frame(matrix(NA,dim(test_pred)[1],1))`
	
	`test_pred_class[test_pred[,"Down"] > 0.5,1]`
	`<- "Down"`

	`test_pred_class[test_pred[,"NoWhere"] > 0.5,1] <- "NoWhere"`
	
	`test_pred_class[test_pred[,"Up"] > 0.5,1] <- "Up"`

Check the accuracy of the forecasts	`test_matrix<- confusionMatrix(test_pred_class[,1],testdir)`
	
	`test_matrix`

Generate trade signals using the same pattern as human psychology	

Traders buy when they anticipate up direction and sell when a down direction is expected .	`signal<- ifelse(test_pred_class =="Up",1,ifelse(test_pred_class`
	
	`=="Down",-1,0))`
	
	`signal`

Results

Neural_network function provides the following output:

```
a 15-4-3 network with 79 weights
options were -
```

The first layer (input layer) -> 15

The second layer (hidden layer) -> 4

Third layer (output layer) -> 3

Weighted parameters -> 79

`Dim(norm_traindji)` function provides the following output:

```
[1] 1006   15
```

Number of columns = number of neurons in the input layer = number of input data features = 15

Predictions obtained:

	Down	NoWhere	Up
2016-01-04	1	0.03919873	0
2016-01-05	1	0.03919873	0
2016-01-06	1	0.03919873	0
2016-01-07	1	0.03919873	0
2016-01-08	1	0.03919873	0
2016-01-11	1	0.03919873	0

Forecast accuracy: 88.4% (model trained on training data)

Level of accuracy: good

Confusion Matrix and Statistics

Prediction	Reference Down	NoWhere	Up
Down	30	4	0
NoWhere	11	101	10
Up	0	4	90

Overall Statistics

Accuracy	: 0.884
95% CI	: (0.8377, 0.9209)
No Information Rate	: 0.436
P-Value [Acc > NIR]	: <2.2e-16
Kappa	: 0.8112
Mcnemar's Test P-Value	: NA

Statistics by Class:

	Class: Down	Class: NoWhere	Class: Up
Sensitivity	0.7317	0.9266	0.9000
Specificity	0.9809	0.8511	0.9733
Pos Pred Value	0.8824	0.8279	0.9574
Neg Pred Value	0.9491	0.9375	0.9359
Prevalence	0.1640	0.4360	0.4000
Detection Rate	0.1200	0.4040	0.3600
Detection Prevalence	0.1360	0.4880	0.3760
Balanced Accuracy	0.8563	0.8888	0.9367

Accuracy of the testing data: 82.1%

```
Confusion Matrix and Statistics

          Reference
Prediction  Down  NoWhere  Up
    Down       0        0   0
    NoWhere    2      101   3
    Up         0       26  41


Overall Statistics

               Accuracy : 0.8208
                 95% CI : (0.7554, 0.8749)
    No Information Rate : 0.7341
    P-Value [Acc > NIR] : 0.004927

                  Kappa : 0.6033
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Down  Class: NoWhere  Class: Up
Sensitivity              0.00000          0.7953     0.9318
Specificity              1.00000          0.8913     0.7984
Pos Pred Value               NaN          0.9528     0.6119
Neg Pred Value           0.98844          0.6119     0.9717
Prevalence               0.01156          0.7341     0.2543
Detection Rate           0.00000          0.5838     0.2370
Detection Prevalence     0.00000          0.6127     0.3873
Balanced Accuracy        0.50000          0.8433     0.8651
```

The trading signals worked out by the preceding analysis are as follows:

```
      matrix.NA..dim.test_pred..1...1.
 [1,]                                1
 [2,]                                1
 [3,]                                1
 [4,]                                1
 [5,]                                0
 [6,]                                0
 [7,]                                0
 [8,]                                0
 [9,]                                0
[10,]                                0
```

3.2.9 Notes: The Heath-Jarrow-Morton (HJM) Framework

The Heath-Jarrow-Morton (1992) framework is used to model the forward interest rate. The HJM framework can be used for pricing bonds and derivatives. It assumes that there is a connection between the drift and the volatility parameters of the forward-rate dynamics in a no-arbitrage world. The Markovian property obtains when the future evolution of assets depend on the present state and not on their past evolution. (For example, the Brownian motion has the Markov property.) HJM models are non-Markovian.

Equations

The zero-coupon bond price at $Z(t;T)$ (discounting factor)

The yield curve at time t:

$$Y(t;T) = -\frac{logZ(t;T)}{T-t}.$$

The forward rate $f(t;T,T+\delta)$ as seen at time t for the period between time T and time $T+\delta$

$$Z(t;T+\delta) = Z(t;T)\exp(-f(t;T,T+\delta)\delta),$$

or

$$f(t;T,T+\delta) = -\frac{lnZ(t;T+\delta)-\ln Z(t;T)}{\delta}.$$

The instantaneous forward rate:

$$F(t;T) \text{ at time } t,$$

$$F(t;T) = \lim_{\delta \to 0} f(t;T,T+\delta).$$

So

$$F(t;T) = -\frac{\partial}{\partial T} lnZ(t;T).$$

Links with $(r;T)$, $Y(t;T)$, and the spot rate $r(t)$

$$Z(t;T) = exp - \left(\int_t^T F(t;s)ds\right),$$
$$Y(t;T) = \frac{1}{T-t}\int_t^T F(t;s)ds,$$
$$r(t) = F(t;T),$$
$$Z(t;T+\delta) = Z(t;T)\frac{1}{(1+\delta F(t;T,T+\delta))},$$

or

$$F(t;T,T+\delta) = \frac{1}{\delta}\left(\frac{Z(t;T)}{Z(t;T+\delta)} - 1\right).$$

So:

$$F(t;T) = -\frac{\partial}{\partial T} lnZ(t;T).$$

In a real world, all zero-coupon bonds evolve according to the following:

$$dZ(t;T) = \mu(t,T)Z(t;T)dt + \sigma(t,T)Z(t;T)dW,$$

Time t evolves but the maturity date T is fixed and

$$\sigma(t,t) = 0.$$

Applying Ito's Lemma[2] gives

[2] You will learn about Ito's Lemma in much greater depth in your upcoming courses.

$$dF(t;T) = \frac{\partial}{\partial T}\left(\frac{1}{2}\sigma^2(t,T) - \mu(t,T)\right)dt - \frac{\partial\sigma(t,T)}{\partial T}dW.$$

This is the forward rate process in a real world.

Spot rate

$$dZ(t;T) = r(t)Z(t;T)dt + \sigma(t,T)Z(t;T)dW.$$

Here $r(t)$ is the spot rate in a risk-neutral world.

Forward rate curve

In the risk-neutral world, the forward rate curve follows:

$$dF(t;T) = m(t,T)dt + v(t,T)dW,$$

where

$$m(t,T) = v(t,T)\int_t^T v(t,s)ds.$$

$P(t,T)$ is the price at time t of a discount bond with principal of \$1 maturing at T.

Wt: represents a vector of past and present values of interest rates and bond prices at time t that are relevant for determining bond price volatilities at that time.

$v(t,T,W_t)$ is the volatility of $P(t,T)$.

$f(t,T_1,T_2)$ is the forward rate as seen at t for the period between T_1 and T_2.

$F(t,T)$ is the instantaneous forward rate as seen at t for a contract maturing at T.

$r(t)$ is the short-term risk-free interest rate at t.

$dz(t)$ is the Wiener process driving term structure movements.

3.2.10 Transcript: Course Review

This brings us to the end of the Econometrics course. Let's do a quick recap of what we've learnt throughout the course.

In Module 1, "Learning R and Stylized Facts of Financial Data", we learned how to use R and studied various empirical regularities of static univariate distributions. We also considered static bivariate distributions to study how the distribution of asset returns can be correlated.

In Module 2, "The Generalized Linear Model", we laid the groundwork of the regression framework that forms the basis from which the more complicated models in the rest of the course could be understood and evaluated.

In Module 3, "Univariate Time Series Modeling", we studied the univariate modeling of the conditional expectation of time series processes. These are necessary tools for the modeling of the expected returns on a specific asset which may be not be well captured by a simple, completely static, unchanging distribution. That is, when returns are persistent (for instance, *weakly* stationary), there is important information in last period's return that will aid in predicting the return of the current period. We showed that it was inappropriate to simply assume that the return every period is from for example, the *same, constant mean* normal distribution, which is typically the simple situation that basic optimal portfolio theory starts with to develop the main ideas. Diversification, the core of portfolio construction, is designed to reduce joint risk. For certain instruments, however, it is inevitable that some persistence in expected returns remain in a portfolio, no matter how well diversified. Hence, we will need the results from univariate time series modeling.

In Module 4, "Univariate Volatility Modeling", we showed that most stock market returns are characterized by persistent volatility clustering. That is, one period of

unusually high volatility tends to be followed by a number of future periods of similarly elevated volatility. We exploited this persistence to build models that could give us the *joint* best prediction of the next period's conditional expectation *and* conditional variance in a given stock return. This is, of course, our best guess, based on current information, of the expected future *return and risk* in investing in a specific asset, and will thus be a part of the optimal portfolio construction.

In Module 5, "Multivariate Modeling", we studied how to model several time series processes simultaneously. In the last part of that module, we studied the multivariate extension of the simultaneous modeling of the conditional expectations, conditional variances *and* conditional covariances. Since the key characteristic necessary for effective diversification of a portfolio is the covariance (or correlation) between a set of assets, this modeling device is an essential tool for optimal portfolio construction.

The approaches in Modules 3 to 5, however, considered the "most likely" outcomes, for instance the central moments of the individual and joint processes. Of key interest to the financial engineer, however, are the probabilities and associated losses related to unusual, or extreme events in some distribution. For this we developed, in Module 6, the results from Extreme Value Theory and applied it to a specific stock or index on an individual level. A typical "stylized fact", however, is that in normal times, stocks, particularly in different sectors of the economy (for example industrial versus consumer goods) are not highly correlated. However, extreme events tend to induce large movements in all assets. Thus, extreme events might be strongly correlated across various usually mostly independent stock returns.

In this module, Module 7, we introduced a new technique. Copulas are the standard approach to modeling interdependence among different time series processes and various techniques in this discipline have been specifically developed to study "joint tail events". That is, cases where extreme events tend to occur simultaneously in two (or more) time series processes.

I hope this module was an enriching experience and that you will have many opportunities to use your knowledge of Econometrics.

References

Berlinger, E. (2015). *Mastering R for Quantitative Finance*. Birmingham: Packt Publishing.

Daróczi, G., Puhle, M. and Berlinger, E. (2013). *Introduction to R for quantitative finance*. Birmingham: Packt Publishing.

Greene, W. (2000). *Econometric analysis*. New York: Prentice-Hall.

Gujarati, D. (2004). *Basic Econometrics*. McGraw-Hill.

Halls-Moore, M. (2017). *Successful Algorithmic Trading*. New York: QuantStart.

Halls-Moore, M. (2017). *Advanced Algorithmic Trading*. New York: QuantStart

Heath, D., Jarrow, R. and Morton, A. (1992). Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Valuation. *Econometrica*, 60(1), pp.77-105.

Heydt, M. (2015). *Mastering Pandas for Finance*. Birmingham: Packt Publishing.

Jeet, P. and Vats, P. *Learning Quantitative Finance with R*. . Birmingham: Packt Publishing.

Kuan, C. and White, H. (1994). Artificial Neural Networks: An Econometric Perspective. *Econometric Reviews*, [online] 13(1), pp.1-91. Available at: https://www.tandfonline.com/doi/abs/10.1080/07474939408800273.

McNeil, A., Frey, R. and Embrechts, P. (2015). *Quantitative risk management*. Princeton, New Jersey: Princeton University Press.

Tattar, P., Ojeda, T., Murphy, S., Bengfort, B. and Dasgupta, A. (2017). *Practical Data Science Cookbook*. Birmingham: Packt Publishing.

White, H. (1989). Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, 1(4), pp.425-464.