

A Statistical Arbitrage Strategy in R

Statistical Arbitrage Strategy

Jacques Joubert

## Background

For those of you who have been following my blog posts for the last 6 months will know that I have taken part in the [Executive Programme in Algorithmic Trading](https://www.quantinsti.com/courses/epat/) offered by QuantInsti.

It’s been a journey and this article serves as a report on my final project focusing on statistical arbitrage, coded in R. This article is a combination of my class notes and my source code.

I uploaded everything to [GitHub](https://github.com/Jackal08/QuantInsti-Final-Project-Statistical-Arbitrage) in order to welcome readers to contribute, improve, use, or work on this project. It will also form part of my [Open Source Hedge Fund project](http://www.quantsportal.com/open-source-hedge-fund/)on my blog QuantsPortal

I would like to say a special thank you to the team at QuantInsti. Thank you for all the revisions of my final project, for going out of your way to help me learn, and the very high level of client services.

## History of Statistical Arbitrage

First developed and used in the mid-1980s by Nunzio Tartaglia’s quantitative group at Morgan Stanly.

* Pair Trading is a “contrarian strategy” designed to harness mean-reverting behavior of the pair ratio
* David Shaw, founder of D.E Shaw & Co, left Morgan Stanley and started his own “Quant” trading firm in the late 1980s dealing mainly in pair trading

## What is Pair Trading?

Statistical arbitrage trading or pairs trading as it is commonly known is defined as trading one financial instrument or a basket of financial instruments – in most cases to create a value neutral basket.

It is the idea that a co-integrated pair is mean reverting in nature. There is a spread between the instruments and the further it deviates from its mean, the greater the probability of a reversal.

Note however that statistical arbitrage is not a risk free strategy. Say for example that you have entered positions for a pair and then the spread picks up a trend rather than mean reverting.

**The Concept**

**Step 1**: Find 2 related securities

Find two securities that are in the same sector / industry, they should have similar market capitalization and average volume traded.

An example of this is Anglo Gold and Harmony Gold.

**Step 2**: Calculate the spread

In the code to follow I used the pair ratio to indicate the spread. It is simply the price of asset A / price asset B.

**Step 3**: Calculate the mean, standard deviation, and z-score of the pair ratio / spread.

**Step 4**: Test for co-integration

In the code to follow I use the Augmented Dicky Fuller Test (ADF Test) to test for co-integration. I set up three tests, each with a different number of observations (120, 90, 60), all three tests have to reject the null hypothesis that the pair is not co-integrated.

**Step 5**: Generate trading signals

Trading signals are based on the z-score, given they pass the test for co-integration. In my project I used a z-score of 1 as I noticed that other algorithms that I was competing with were using very low parameters. (I would have preferred a z-score of 2, as it better matches the literature, however it is less profitable)

**Step 6**: Process transactions based on signals

**Step 7**: Reporting

# R markdown for my project

## Import packages and set directory

The first step is always to import the packages needed.

#Imports  
require(tseries)  
require(urca) #Used for the ADF Test  
require(PerformanceAnalytics)

This strategy will be run on shares listed on the Johannesburg Stock Exchange (JSE); because of this I wont be using the quantmod package to pull data from yahoo finance, instead I have already gotten and cleaned the data that I stored in a SQL database and moved to csv files on the Desktop.

I added all the pairs used in the strategy to a folder which I now set to be the working directory.

##Change this to match where you stored the csv files folder name FullList  
setwd("~/R/QuantInsti-Final-Project-Statistical-Arbitrage/database/FullList")

## Functions that will be called from within other functions (No user interaction)

Next: Create all the functions that will be needed. The functions below will be called from within other functions so you dont need to worry about the arguments.

### AddColumns

The AddColumns function is used to add columns to the dataframe that will be needed to store variables.

#Add Columns to csvDataframe  
AddColumns <- function(csvData){  
 csvData$spread <- 0  
 csvData$adfTest <- 0  
 csvData$mean <- 0  
 csvData$stdev <- 0  
 csvData$zScore <- 0  
 csvData$signal <- 0  
 csvData$BuyPrice <- 0  
 csvData$SellPrice <- 0  
 csvData$LongReturn <- 0  
 csvData$ShortReturn <- 0  
 csvData$Slippage <- 0  
 csvData$TotalReturn <- 0  
 csvData$TransactionRatio <- 0  
 csvData$TradeClose <- 0  
 return(csvData)  
}

### PrepareData

The PrepareData function calculates the pair ratio and the log10 prices of the pair. It also calls the AddColumns funtion within it.

PrepareData <- function(csvData){  
 #Calculate the Pair Ratio  
 csvData$pairRatio <- csvData[,2] / csvData[,3]  
   
 #Calculate the log prices of the two time series  
 csvData$LogA <- log10(csvData[,2])  
 csvData$LogB <- log10(csvData[,3])  
   
 #Add columns to the DF  
 csvData <- AddColumns(csvData)  
   
 #Make sure that the date column is not read in as a vector of characters  
 csvData$Date <- as.Date(csvData$Date)  
   
 return(csvData)  
}

### GenerateRowValue

The GenerateRowValue function Calculates the mean, standard deviation and the z-score for a given row in the dataframe.

#Calculate mean, stdDev, and z-score for the given Row [end]  
GenerateRowValue <- function(begin, end, csvData){  
 average <- mean(csvData$spread[begin:end])  
 stdev <- sd(csvData$spread[begin:end])  
   
 csvData$mean[end] <- average  
 csvData$stdev[end] <- stdev  
 csvData$zScore[end] <- (csvData$spread[end]-average)/stdev  
   
 return(csvData)  
   
}

### GenerateSignal

The GenerateSignal function creates a long, short, or close signal based on the z-score. You can manually change the z-score. I have set it to 1 and -1 for entry signals and any z-score between 0.5 and -0.5 will create a close/exit signal.

GenerateSignal <- function(counter, csvData){  
 #Trigger and close represent the entry and exit zones (value refers to the z-score value)  
 trigger <- 1  
 close <- 0.5  
   
 currentSignal <- csvData$signal[counter]  
 prevSignal <- csvData$signal[counter-1]  
   
 #Set trading signal for the given [end] row  
 if(csvData$adfTest[counter] == 1)  
 {  
 #If there is a change in signal from long to short then you must allow for the   
 #current trade to first be closed  
 if(currentSignal == -1 && prevSignal == 1)  
 csvData$signal[counter] <- 0  
 else if(currentSignal == 1 && prevSignal == -1)  
 csvData$signal[counter] <- 0  
   
 #Create a long / short signal if the current z-score is larger / smaller than the trigger value  
 #(respectively)  
 else if(csvData$zScore[counter] > trigger)  
 csvData$signal[counter] <- -1  
 else if (csvData$zScore[counter] < -trigger)  
 csvData$signal[counter] <- 1  
   
 #Close the position if z-score is beteween the two "close" values  
 else if (csvData$zScore[counter] < close && csvData$zScore[counter] > -close)  
 csvData$signal[counter] <- 0  
 else   
 csvData$signal[counter] <- prevSignal  
 }  
 else   
 csvData$signal[counter] <- 0  
   
 return(csvData)  
}

### GenerateTransactions

The GenerateTransactions function is responsible for setting the entry and exit prices for the respective long and short positions needed to create a pair.

Note: QuantInsti taught us a very specific way of backtesting a trading strategy. They used excel to teach strategies and when I coded this strategy I used a large part of the excel methodology.

Going forward however I would explore other ways of storing variabels. One of the great things about this method is that you can pull the entire dataframe and analys why a trade was made and all the details pertaining to it.

#Transactions based on trade signal  
#Following the framework set out initially by QuantInsti (Note: this can be coded better)   
GenerateTransactions <- function(currentSignal, prevSignal, end, csvData){  
 #In a pair trading strategy you need to go long one share and short the other  
 #and then reverse the transaction when you close  
   
 ##First Leg of the trade (Set Long position)  
 #If there is no change in signal  
 if(currentSignal == 0 && prevSignal == 0)  
 {  
 csvData$BuyPrice[end] <- 0   
 csvData$TransactionRatio[end]<-0  
 }  
 else if(currentSignal == prevSignal)  
{ csvData$BuyPrice[end] <- csvData$BuyPrice[end-1]   
 csvData$TransactionRatio[end]<-csvData$TransactionRatio[end-1]  
}   
 #If the signals point to a new trade  
 #Short B and Long A  
 else if(currentSignal == 1 && currentSignal != prevSignal)  
 csvData$BuyPrice[end] <- csvData[end, 2]   
 #Short A and Long B  
 else if(currentSignal == -1 && currentSignal != prevSignal){  
 csvData$BuyPrice[end] <- csvData[end, 3] \* csvData$pairRatio[end]  
 transactionPairRatio <<- csvData$pairRatio[end]  
 csvData$TransactionRatio[end]<- transactionPairRatio  
 }  
   
 #Close trades  
 else if(currentSignal == 0 && prevSignal == 1)  
 csvData$BuyPrice[end] <- csvData[end, 2]   
 else if(currentSignal == 0 && prevSignal == -1)  
 {csvData$TransactionRatio[end] = csvData$TransactionRatio[end-1]  
 csvData$BuyPrice[end] <- csvData[end, 3] \* csvData$TransactionRatio[end]  
 }  
   
   
 ##Second Leg of the trade (Set Short position)  
 ##Set Short Prices if there is no change in signal  
 if(currentSignal == 0 && prevSignal == 0)  
 csvData$SellPrice[end] <- 0   
 else if(currentSignal == prevSignal)  
 csvData$SellPrice[end] <- csvData$SellPrice[end-1]   
   
 #If the signals point to a new trade  
 else if(currentSignal == 1 && currentSignal != prevSignal){  
 csvData$SellPrice[end] <- csvData[end, 3] \* csvData$pairRatio[end]  
 transactionPairRatio <<- csvData$pairRatio[end]  
 csvData$TransactionRatio[end]<- transactionPairRatio  
 }  
 else if(currentSignal == -1 && currentSignal != prevSignal)  
 csvData$SellPrice[end] <- csvData[end, 2]   
   
 #Close trades  
 else if(currentSignal == 0 && prevSignal == 1){  
 csvData$TransactionRatio[end] = csvData$TransactionRatio[end-1]  
 csvData$SellPrice[end] <- csvData[end, 3] \* csvData$TransactionRatio[end]  
 }  
 else if(currentSignal == 0 && prevSignal == -1)  
 csvData$SellPrice[end] <- csvData[end, 2]   
   
 return(csvData)  
}

### GetReturnsDaily

GetReturnsDaily calculates the daily returns on each position and then calculates the total returns and adds slippage.

#Calculate daily returns generated   
#Add implementation shortfall / slippage at close of trade  
GetReturnsDaily <- function(end, csvData, slippage){  
 #Calculate the returns generated on each leg of the deal (the long and the short position)  
 #Long leg of the trade  
 if(csvData$signal[end-1]>0){csvData$LongReturn[end] <- log(csvData[end,2] / csvData[end-1,2])}  
 else  
 if(csvData$signal[end-1]<0){csvData$LongReturn[end] <- log(csvData[end,3] / csvData[end-1,3])\*csvData$TransactionRatio[end]}   
  
 #Short Leg of the trade  
 if(csvData$signal[end-1]>0){csvData$ShortReturn[end] <- -log(csvData[end,3] / csvData[end-1,3])\*csvData$TransactionRatio[end]}  
 else  
 if(csvData$signal[end-1]<0){csvData$ShortReturn[end] <- -log(csvData[end,2] / csvData[end-1,2])}   
  
 #Add slippage  
 if(csvData$signal[end] == 0 && csvData$signal[end-1] != 0)  
 {  
 csvData$Slippage[end] <- slippage  
 csvData$TradeClose[end] <-1  
 }  
 #If a trade was closed then calculate the total return  
 csvData$TotalReturn[end] <- ((csvData$ShortReturn[end] + csvData$LongReturn[end]) / 2) + csvData$Slippage[end]  
   
 return(csvData)  
}

### GenerateReports

The next two arguments are used to generate reports. A report includes the following: Charting: 1. An Equity curve 2. Drawdown curve 3. Daily returns bar chart

Statistics: 1. Annual Retruns 2. Annualized Sharpe Ratio 3. Maximum Drawdown

Table: 1. Top 5 drawdowns and their duration

Note: If you have some extra time then you can further break this function down into smaller functions inorder to reduce the lines of code and improve userbility. Less code = Less Bugs

#Returns an equity curve, annualized return, annualized sharpe ratio, and max drawdown  
GenerateReport <- function(pairData, startDate, endDate){  
 #Subset the dates   
 returns <- xts(pairData$TotalReturn, as.Date(pairData$Date))  
 returns <- returns[paste(startDate,endDate,sep="::")]  
 #Plot  
 charts.PerformanceSummary(returns)  
   
 #Metrics  
 print(paste("Annual Returns: ",Return.annualized(returns)))  
 print(paste("Annualized Sharpe: " ,SharpeRatio.annualized(returns)))  
 print(paste("Max Drawdown: ",maxDrawdown(returns)))  
  
 pairDataSub= pairData[pairData$TradeClose==1,]  
   
 returns\_sub <- xts(pairDataSub$TotalReturn, as.Date(pairDataSub$Date))  
 returns\_sub <- returns\_sub[paste(startDate,endDate,sep="::")]   
 #var returns = xts object  
 totalTrades <- 0  
 positiveTrades <- 0  
 profitsVector <- c()  
 lossesVector <- c()  
   
 #loop through the data to find the + & - trades and total trades  
 for(i in returns\_sub){  
 if(i != 0){  
 totalTrades <- totalTrades + 1  
 if(i > 0){  
 positiveTrades <- positiveTrades + 1  
 profitsVector <- c(profitsVector, i)  
 }  
 else if (i < 0){  
 lossesVector <- c(lossesVector, i)  
 }  
 }  
 }  
   
 #Print the results to the console  
 print(paste("Total Trades: ", totalTrades))  
 print(paste("Success Rate: ", positiveTrades/totalTrades))  
 print(paste("PnL Ratio: ", mean(profitsVector)/mean(lossesVector\*-1)))  
 print(table.Drawdowns(returns))  
   
}  
  
GenerateReport.xts <- function(returns, startDate = '2005-01-01', endDate = '2015-11-23'){  
 #Metrics  
 returns <- returns[paste(startDate,endDate,sep="::")]  
 charts.PerformanceSummary(returns)  
 print(paste("Annual Returns: ",Return.annualized(returns)))  
 print(paste("Annualized Sharpe: " ,SharpeRatio.annualized(returns)))  
 print(paste("Max Drawdown: ",maxDrawdown(returns)))  
 print(table.Drawdowns(returns))  
   
}

## Functions that the user will pass parameters to

The next two functions are the only functions that the user should fiddle with.

### BacktestPair

BacktestPair is used when you want to run a backtest on a trading pair (the pair is passed in via the csv file)

Functions arguments:

* pairData = the csv file date
* mean = the number of observations used to calculate the mean of the spread.
* slippage = the amount of basis points that act as brokerage as well as slippage
* adfTest = a boolean value - if the backtest should test for co-integration
* criticalValue = Critical Value used in the ADF Test to test for co-integration
* generateReport = a boolean value - if a report must be generated

#The function that will be called by the user to backtest a pair  
BacktestPair <- function(pairData, mean = 35, slippage = -0.0025, adfTest = TRUE, criticalValue = -2.58,  
 startDate = '2005-01-01', endDate = '2014-11-23', generateReport = TRUE){  
 # At 150 data points  
 # Critical value at 1% : -3.46  
 # Critical value at 5% : -2.88  
 # Critical value at 10% : -2.57  
   
 #Prepare the initial dataframe by adding columns and pre calculations  
 pairData <- PrepareData(pairData)  
   
 #Itterate through each day in the time series  
 for(i in 1:length(pairData[,2])){  
 #For each day after the amount of days needed to run the ADF test  
 if(i > 130){  
 begin <- i - mean + 1  
 end <- i  
   
 #Calculate Spread  
 spread <- pairData$pairRatio[end]  
 pairData$spread[end] <- spread  
   
 #ADF Test   
 #120 - 90 - 60   
 if(adfTest == FALSE){  
 pairData$adfTest[end] <- 1   
 }  
 else {  
 if(adf.test(pairData$spread[(i-120):end], k = 1)[1] <= criticalValue){  
 if(adf.test(pairData$spread[(i-90):end], k = 1)[1] <= criticalValue){  
 if(adf.test(pairData$spread[(i-60):end], k = 1)[1] <= criticalValue){  
 #If co-integrated then set the ADFTest value to true / 1  
 pairData$adfTest[end] <- 1   
 }  
 }  
 }  
 }  
 #Calculate the remainder variables needed  
 if(i >= mean){  
 #Generate Row values  
 pairData <- GenerateRowValue(begin, end, pairData)  
 #Generate the Signals  
 pairData <- GenerateSignal(i, pairData)  
   
 currentSignal <- pairData$signal[i]  
 prevSignal <- pairData$signal[i-1]  
   
 #Generate Transactions  
 pairData <- GenerateTransactions(currentSignal, prevSignal, i, pairData)  
   
 #Get the returns with added slippage  
 pairData <- GetReturnsDaily(i, pairData, slippage)  
  
 }  
 }  
 }  
   
 if(generateReport == TRUE)  
 GenerateReport(pairData, startDate, endDate)  
  
 return(pairData)  
}

### BacktestPortfolio

BacktestPortfolio accepts a vector of csv files and then generates an equaly weighted portfolio.

Functions arguments:

* names = an attomic vector of csv file names, example: c('DsyLib.csv', 'OldSanlam.csv')
* mean = the number of observations used to calculate the mean of the spread.
* leverage = how much leverage you want to apply to the portfolio

#An equally weighted portfolio of shares  
BacktestPortfolio <- function(names, mean = 35,leverage = 1, startDate = '2005-01-01', endDate = '2015-11-23'){  
 ##Itterates through all the pairs and backtests each one  
 ##stores the data in a list of numerical vectors  
 returns.list <- list()  
 counter <- F  
 ticker <- 1  
 for (name in names){  
 #A notification to let you know how far it is  
 print(paste(ticker, " of ", length(names)))  
 ticker <- ticker + 1  
   
 #Run the backtest on the pair  
 data <- read.csv(name)   
 BackTest.df <- BacktestPair(data, mean, generateReport = FALSE)  
   
 #Store the dates in a seperate vector  
 if (counter == F){  
 dates <<- as.Date(BackTest.df$Date)  
 counter <- T  
 }  
   
 #Append to list  
 returns.list <- c(returns.list, list(BackTest.df[,18]))  
 }  
   
 ##Aggregates the returns for each day and then calculates the average for each day  
 total.returns <- c()  
 for (i in 1:length(returns.list)){  
 if(i == 1)  
 total.returns = returns.list[[i]]  
 else  
 total.returns = total.returns + returns.list[[i]]  
 }  
   
 total.returns <- total.returns / length(returns.list)  
   
 ##Generate a report for the portfolio  
 returns <- xts(total.returns \* leverage, dates)  
 GenerateReport.xts(returns, startDate, endDate)  
   
 return(returns)  
}

# Running Backtests

Now we can start testing strategies using our code.

## Pure arbitrage on the JSE

WHen starting this project the main focus was on using statistical arbitrage to find pairs that were co-integrated and then to trade those, however I very quickly realised that the same code could be used to trade shares that had both its primary listing as well as access to its secondary listing on the same exchange.

If both listings are found on the same exchange, it opens the door for a pure arbitrage strategy due to both listings refering to the same asset. Therefore you dont need to test for co-integration.

There are two very obvious examples on the JSE.

### First Example Investec:

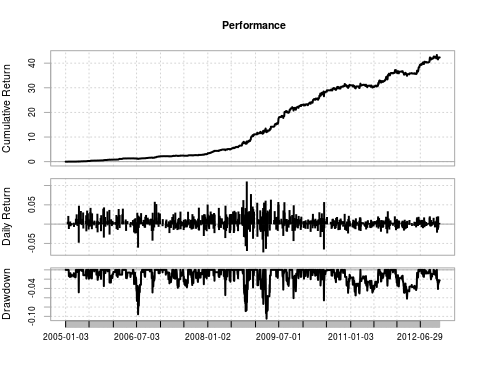
Primary = Investec Ltd : Secondary = Investec PLC

#### Investec In-Sample Test (2005-01-01 - 2012-11-23)

Test the following parameters

* The Investec ltd / plc pair
* mean = 35
* Set adfTest = F (Dont test for co-integration)
* Leverage of x3

#Investec  
leverage <- 3  
data <- read.csv('Investec.csv')   
investec <- BacktestPair(data, 35, generateReport = F, adfTest = F)   
  
#Format to an xts object and pass to GenerateReport.xts()  
investec.returns <- xts(investec[,18] \* leverage, investec$Date)  
GenerateReport.xts(investec.returns, startDate = '2005-01-01', endDate = '2012-11-23')

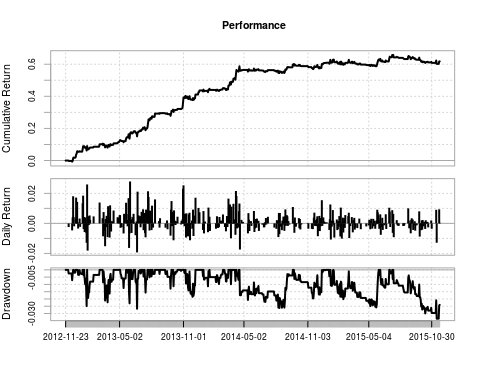


## [1] "Annual Returns: 0.619853087807437"  
## [1] "Annualized Sharpe: 3.29778431709924"  
## [1] "Max Drawdown: 0.105016628973292"  
## From Trough To Depth Length To Trough Recovery  
## 1 2009-03-19 2009-03-25 2009-05-04 -0.1050 28 5 23  
## 2 2006-06-08 2006-07-13 2006-08-14 -0.0955 46 25 21  
## 3 2008-10-03 2008-10-17 2008-10-24 -0.0887 16 11 5  
## 4 2009-03-02 2009-03-02 2009-03-06 -0.0733 5 1 4  
## 5 2008-10-27 2008-10-27 2008-11-05 -0.0697 8 1 7

#### Investec Out-of-Sample Test (2012-11-23 - 2015-11-23)

Note: if you increase the slippage, you will very quickly kiss profits good bye.

GenerateReport.xts(investec.returns, startDate = '2012-11-23', endDate = '2015-11-23')



## [1] "Annual Returns: 0.1754103210963"  
## [1] "Annualized Sharpe: 2.20385429706265"  
## [1] "Max Drawdown: 0.0335642102186873"  
## From Trough To Depth Length To Trough Recovery  
## 1 2015-07-10 2015-11-13 <NA> -0.0336 96 89 NA  
## 2 2013-06-18 2013-06-21 2013-07-01 -0.0267 10 4 6  
## 3 2014-04-16 2014-08-13 2014-09-19 -0.0262 107 80 27  
## 4 2015-01-20 2015-05-25 2015-06-01 -0.0258 91 86 5  
## 5 2013-01-18 2013-01-24 2013-01-25 -0.0249 6 5 1

### Second Example Mondi:

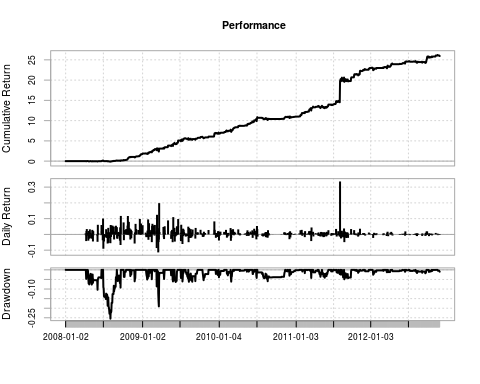
Primary = Mondi Ltd : Secondary = Mondi PLC

#### Mondi In-Sample Test (2008-01-01 - 2012-11-23)

Test the following parameters

* The Mondi ltd / plc pair
* mean = 35
* Set adfTest = F (Dont test for co-integration)
* Leverage of x3

data <- read.csv('mondi.csv')   
mondi <- BacktestPair(data, 35, generateReport = F, adfTest = F)  
  
mondi.returns <- xts(mondi[,18] \* leverage, mondi$Date)  
GenerateReport.xts(mondi.returns, startDate = '2008-01-01', endDate = '2012-11-23')

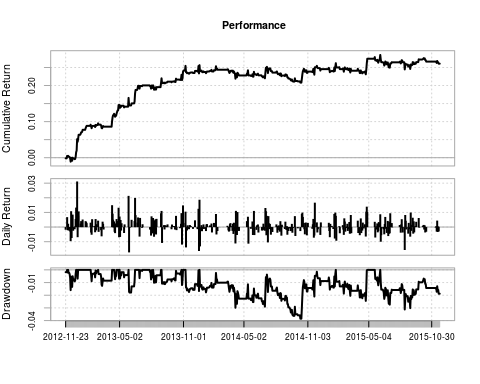


## [1] "Annual Returns: 0.973552250431717"  
## [1] "Annualized Sharpe: 2.88672185296756"  
## [1] "Max Drawdown: 0.254688711989788"  
## From Trough To Depth Length To Trough Recovery  
## 1 2008-07-01 2008-08-01 2008-09-01 -0.2547 45 24 21  
## 2 2009-03-11 2009-03-18 2009-04-08 -0.1906 21 6 15  
## 3 2008-04-16 2008-06-03 2008-06-23 -0.1040 45 32 13  
## 4 2008-09-02 2008-09-17 2008-09-18 -0.0926 13 12 1  
## 5 2009-03-09 2009-03-09 2009-03-10 -0.0864 2 1 1

#### Mondi Out-of-Sample Test (2012-11-23 - 2015-11-23)

Note: In all of my testing I found that the further down the timeline my data was, the harder it was to make profits on end of day data. I tested this same strategy on intradaydata and it has a higher return profile.

GenerateReport.xts(mondi.returns, startDate = '2012-11-23', endDate = '2015-11-23')



## [1] "Annual Returns: 0.0809094579019469"  
## [1] "Annualized Sharpe: 1.25785312960412"  
## [1] "Max Drawdown: 0.0385234269750542"  
## From Trough To Depth Length To Trough Recovery  
## 1 2013-12-19 2014-10-13 2015-01-26 -0.0385 273 202 71  
## 2 2015-06-05 2015-08-14 <NA> -0.0313 120 49 NA  
## 3 2015-01-27 2015-04-22 2015-04-28 -0.0245 63 60 3  
## 4 2013-05-29 2013-05-30 2013-06-14 -0.0179 13 2 11  
## 5 2013-11-08 2013-11-18 2013-12-18 -0.0175 28 7 21

# Statistical Arbitrage on the JSE

Next we will look at a pair trading strategy.

Typically a pair consits of 2 shares that:

* Share a market sector
* Have a similar market cap
* Similar business model and clients
* Are co-integrated

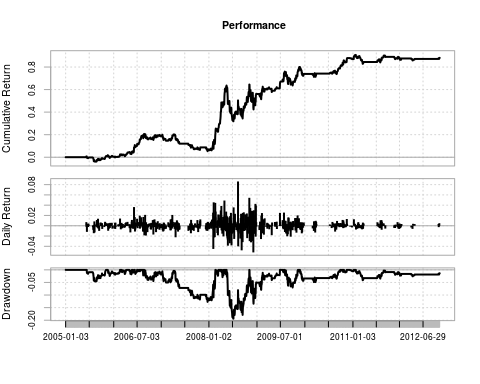
In all of the portfolios below I use 3x leverage

## Contruction Portfolio

### In-sample test (2005-01-01 - 2012-11-01)

names <- c('GroupMR.csv', 'GroupPPC.csv', 'GroupAVENGE.csv', 'GroupWHBO.csv',   
 'mrppc.csv', 'mravenge.csv')  
  
ReturnSeries <- BacktestPortfolio(names, startDate = '2005-01-01', endDate = '2012-11-01', leverage = 3)

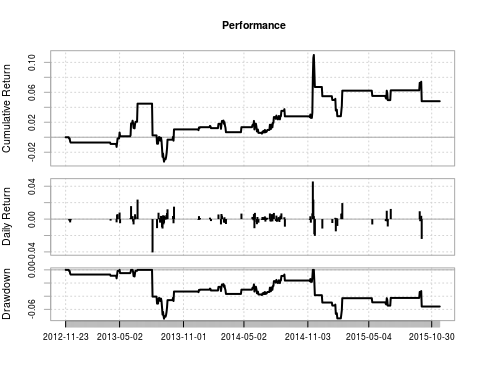
## [1] "1 of 6"  
## [1] "2 of 6"  
## [1] "3 of 6"  
## [1] "4 of 6"  
## [1] "5 of 6"  
## [1] "6 of 6"



## [1] "Annual Returns: 0.0848959306632411"  
## [1] "Annualized Sharpe: 0.733688101181479"  
## [1] "Max Drawdown: 0.193914686702112"  
## From Trough To Depth Length To Trough Recovery  
## 1 2008-05-19 2008-07-08 2008-11-03 -0.1939 119 36 83  
## 2 2008-11-04 2008-12-03 2009-06-29 -0.1345 160 22 138  
## 3 2006-08-25 2007-12-19 2008-02-19 -0.1272 372 331 41  
## 4 2009-08-04 2009-10-01 2009-11-10 -0.0701 69 41 28  
## 5 2009-11-25 2010-03-10 2010-09-29 -0.0486 211 73 138

### Out-of-sample test (2012-11-23 - 2015-11-23)

GenerateReport.xts(ReturnSeries, startDate = '2012-11-23', endDate = '2015-11-23')



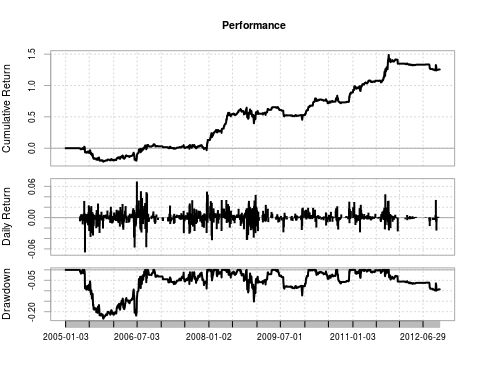
## [1] "Annual Returns: 0.0159094762396512"  
## [1] "Annualized Sharpe: 0.268766025866724"  
## [1] "Max Drawdown: 0.0741426720423424"  
## From Trough To Depth Length To Trough Recovery  
## 1 2013-08-05 2013-09-06 2014-11-17 -0.0741 322 24 298  
## 2 2014-11-20 2015-01-29 <NA> -0.0737 253 47 NA  
## 3 2012-11-30 2013-04-23 2013-05-02 -0.0129 102 96 6  
## 4 2013-06-10 2013-06-13 2013-06-24 -0.0100 10 4 6  
## 5 2013-05-03 2013-05-03 2013-06-04 -0.0050 23 1 22

## Insurance Portfolio

### In-sample test (2005-01-01 - 2012-11-01)

names <- c('DiscLib.csv', 'DiscMMI.csv', 'DiscSanlam.csv', 'LibMMI.csv', 'MMIOld.csv',  
 'MMISanlam.csv', 'OldSanlam.csv')  
  
ReturnSeries <- BacktestPortfolio(names, startDate = '2005-01-01', endDate = '2012-11-01', leverage = 3)

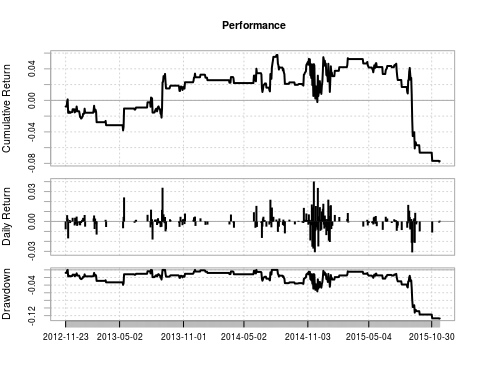
## [1] "1 of 7"  
## [1] "2 of 7"  
## [1] "3 of 7"  
## [1] "4 of 7"  
## [1] "5 of 7"  
## [1] "6 of 7"  
## [1] "7 of 7"



## [1] "Annual Returns: 0.110600985165525"  
## [1] "Annualized Sharpe: 0.791920916349154"  
## [1] "Max Drawdown: 0.233251846760865"  
## From Trough To Depth Length To Trough Recovery  
## 1 2005-05-26 2005-10-14 2006-08-31 -0.2333 318 100 218  
## 2 2008-10-15 2008-12-05 2009-04-30 -0.1513 134 38 96  
## 3 2009-06-10 2009-12-10 2010-01-29 -0.1223 162 129 33  
## 4 2011-10-04 2012-10-09 <NA> -0.0991 267 249 NA  
## 5 2006-11-08 2007-12-11 2007-12-14 -0.0894 277 274 3

### Out-of-sample test (2012-11-23 - 2015-11-23)

GenerateReport.xts(ReturnSeries, startDate = '2012-11-23', endDate = '2015-11-23')



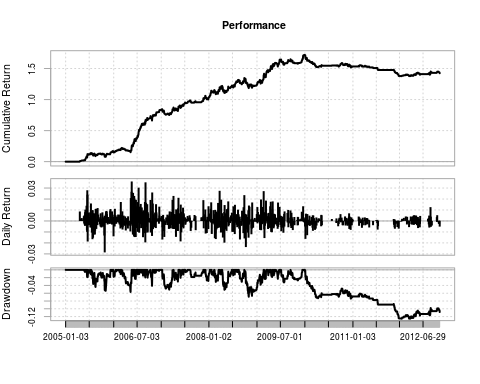
## [1] "Annual Returns: -0.0265926093350092"  
## [1] "Annualized Sharpe: -0.319582293135835"  
## [1] "Max Drawdown: 0.128061204573991"  
## From Trough To Depth Length To Trough Recovery  
## 1 2014-08-08 2015-11-20 <NA> -0.1281 326 324 NA  
## 2 2012-11-28 2013-05-13 2013-07-31 -0.0393 167 111 56  
## 3 2014-06-10 2014-06-26 2014-07-23 -0.0284 31 12 19  
## 4 2013-08-01 2013-08-30 2013-09-03 -0.0255 23 21 2  
## 5 2013-09-11 2013-10-22 2013-12-04 -0.0209 60 29 31

## General Retail Portfolio

### In-sample test (2005-01-01 - 2012-11-01)

names <- c('Wooltru.csv', 'WoolMr.csv', 'WoolTFG.csv', 'TRUMR.csv', 'TruTFG.csv', 'MRTFG.csv')  
  
ReturnSeries <- BacktestPortfolio(names, startDate = '2005-01-01', endDate = '2012-11-01', leverage = 3)

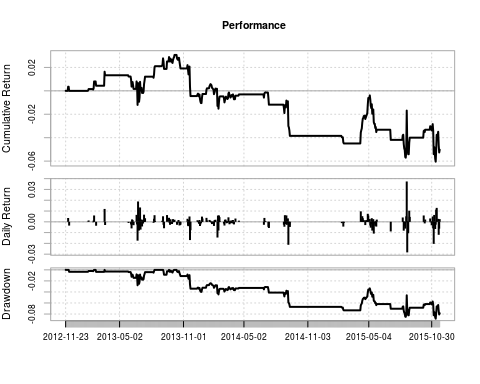
## [1] "1 of 6"  
## [1] "2 of 6"  
## [1] "3 of 6"  
## [1] "4 of 6"  
## [1] "5 of 6"  
## [1] "6 of 6"



## [1] "Annual Returns: 0.120956981644048"  
## [1] "Annualized Sharpe: 1.4694780839876"  
## [1] "Max Drawdown: 0.125406256082082"  
## From Trough To Depth Length To Trough Recovery  
## 1 2010-01-05 2012-01-17 <NA> -0.1254 705 504 NA  
## 2 2008-09-29 2008-10-29 2009-02-20 -0.0690 101 23 78  
## 3 2006-03-06 2006-05-15 2006-05-23 -0.0568 52 46 6  
## 4 2005-07-18 2005-11-01 2005-12-06 -0.0538 101 76 25  
## 5 2008-04-11 2008-04-29 2008-06-26 -0.0512 51 12 39

### Out-of-sample test (2012-11-23 - 2015-11-23)

GenerateReport.xts(ReturnSeries, startDate = '2012-11-23', endDate = '2015-11-23')



## [1] "Annual Returns: -0.0171898953593881"  
## [1] "Annualized Sharpe: -0.336265418351652"  
## [1] "Max Drawdown: 0.0884145115767888"  
## From Trough To Depth Length To Trough Recovery  
## 1 2013-10-15 2015-11-11 <NA> -0.0884 528 519 NA  
## 2 2013-03-18 2013-06-24 2013-08-12 -0.0279 100 66 34  
## 3 2013-09-05 2013-09-06 2013-09-20 -0.0088 12 2 10  
## 4 2013-09-23 2013-10-02 2013-10-08 -0.0049 11 7 4  
## 5 2013-02-20 2013-02-20 2013-03-15 -0.0037 18 1 17

## Conclusion:

At the end of all my testing, and trust me – there is a lot more testing I did than what is in this report, I came to the conclusion that the Pure Arbitrage Strategy has great hope in being used as a strategy using real money, but the Pair Trading Strategy on portfolios of stocks in a given sector is strained and not likely to be used in production in its current form.

There are many things that I think could be added to improve the performance. Going forward I will investigate using Kalman filters.

### More on the Pure Arbitrage Trading Strategy:

I have only found two shares that have duel listings on the same exchange; this means that we can’t allocate large sums of money to the strategy as it will have a high market impact, however we could use multiple exchanges and increase the number of shares used.

### More on the Pair Trading Strategy:

1. The number of observations used in the ADF Tests are largely to blame. The problem is that a test for co-integration has to be done in order to make a claim for statistical arbitrage, however by using 120, 90, and 60 as parameters to the three tests, it is very difficult to find pairs that match the criteria and that will continue in this form for the near future. (Kalman filtering may be useful here)
2. I haven’t spent a lot of time changing the different parameters like the number of observations in the mean calculation. (This requires further exploration)
3. From the above sector portfolios, we can see that the early years are very profitable but the further down the timeline we go, the lower returns get. I have spoken to a few people in the industry as well as my friends doing stat arb projects at the University of Cape Town, the local lore has it that in 2009 Goldman switched on their stat arb package, in regards to the JSE listed securities.
4. The same is noticed with other portfolios that I didn’t include in this report but is in the R Code file.
5. I believe that this is due to large institutions using the same bread and butter strategy. You will note (if you spend enough time testing all the strategies) that in 2009 there seems to be a sudden shift in the data to lower returns.
6. I feel that the end of day data I am using is limiting me and if I were to test the strategy on intraday data then profits would be higher. (I ran one test on intraday data on Mondi and the results were much higher, but I am still to test it on sector portfolios)
7. This is one of the simpler statistical arbitrage strategies and I believe that if we were to improve on the way we calculate the spread and change some of the entry and exit rules, the strategy would become more profitable.

If you made it to the end of this article, I thank you and hope that it added some value. This is the first time that I am using Github, so I am looking forward to see if there are any new contributors to the project.

Github repository: [*https://github.com/Jackal08/QuantInsti-Final-Project-Statistical-Arbitrage*](https://github.com/Jackal08/QuantInsti-Final-Project-Statistical-Arbitrage)