Statistical Arbitrage

This is sourced from STAD70 course practice questions and sample R code taught by professor Sotos. If you have any questions/concerns/comments feel free to email me: cristal.wang111@gmail.com (mailto:cristal.wang111@gmail.com).

1.Pairs Trading and Cointegration Analysis

1.1.Get Historical Stock Data

```
library(tseries)
```

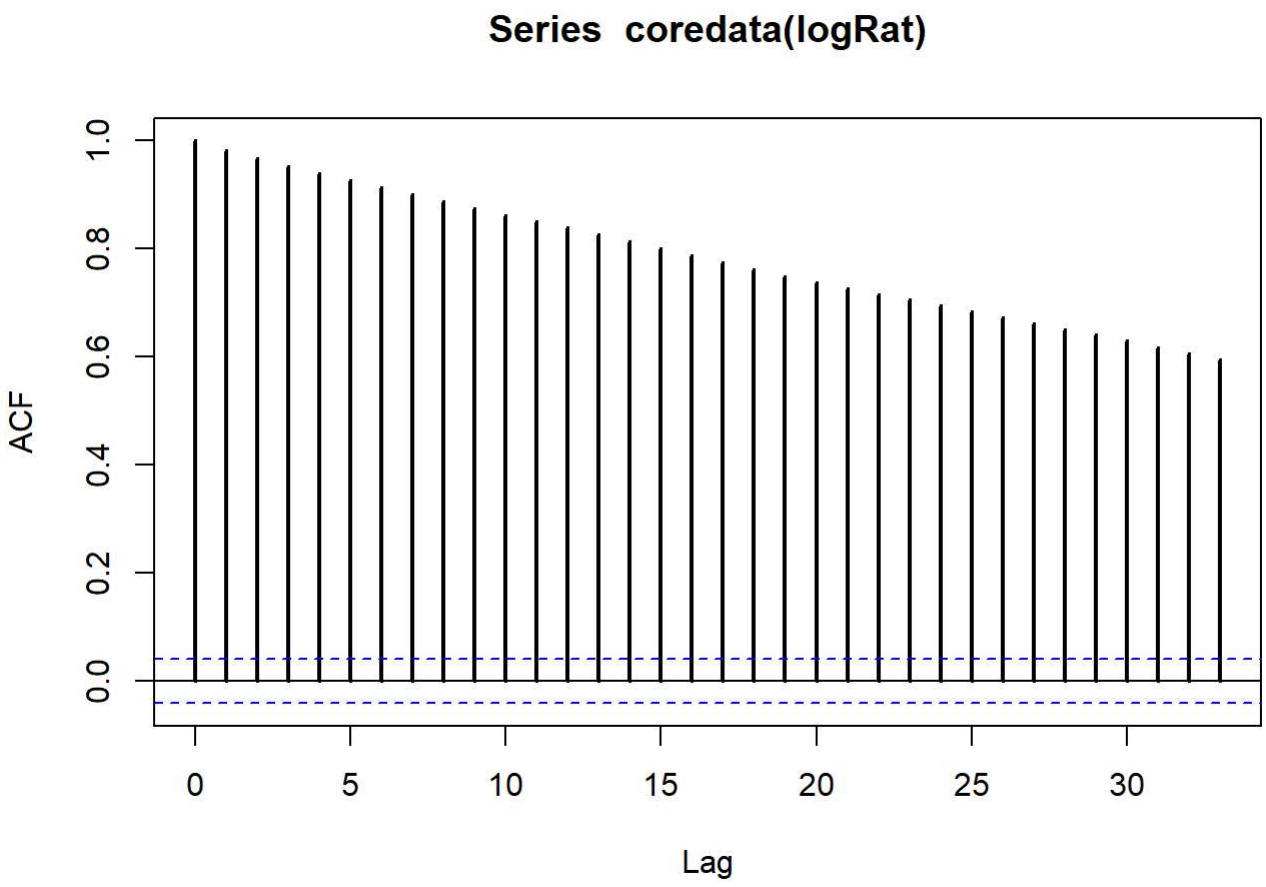
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

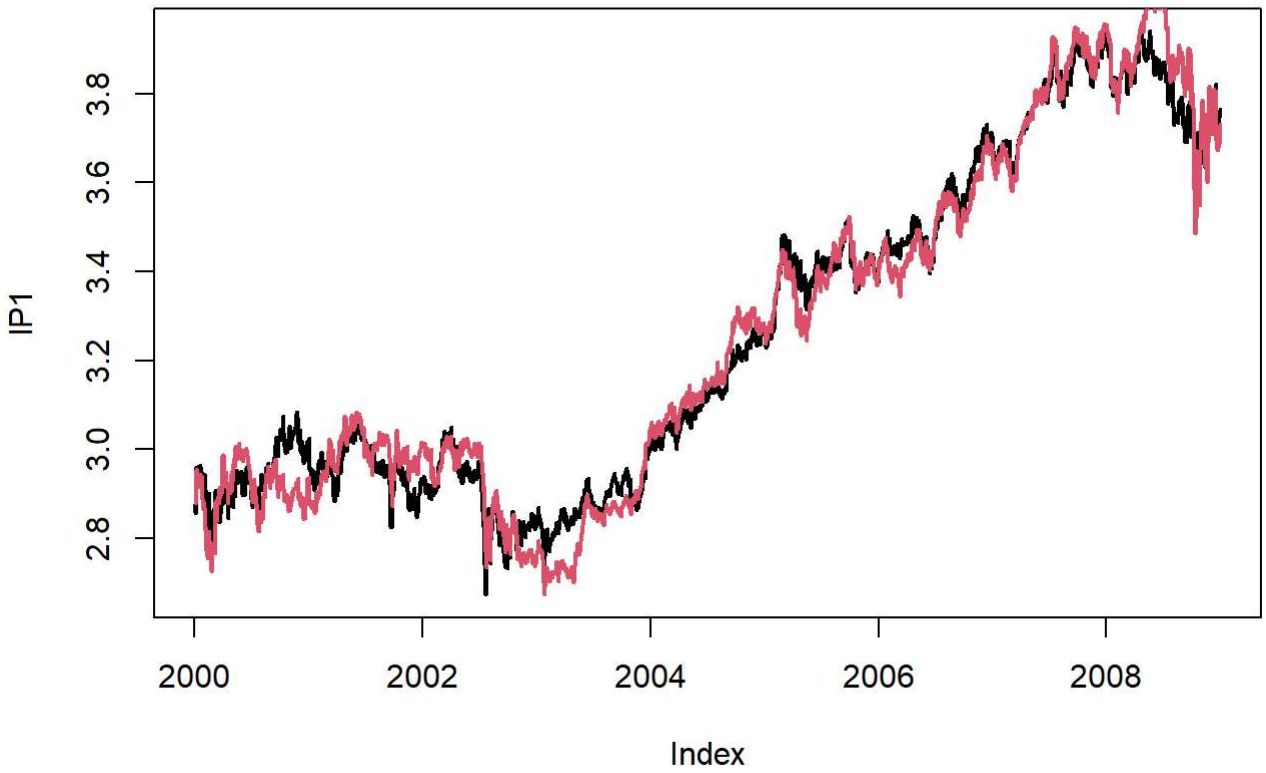
```
start.date=as.Date("2000-01-01"); end.date=as.Date("2008-12-31")
P1=get.hist.quote(instrument = "XOM", start = start.date, end=end.date, quote = "AdjClos
e", retclass = "zoo", quiet = T)
P2=get.hist.quote(instrument = "CVX", start = start.date, end=end.date, quote = "AdjClos
e", retclass = "zoo", quiet = T)
P=merge(P1,P2, all=FALSE); P1=P[,1]; P2=P[,2]
lP1=log(P1); lP2=log(P2)
logRat= lP1-lP2 # log-ratio of prices
acf(coredata(logRat), lwd=2) # log-ratio autocorrelation function
```



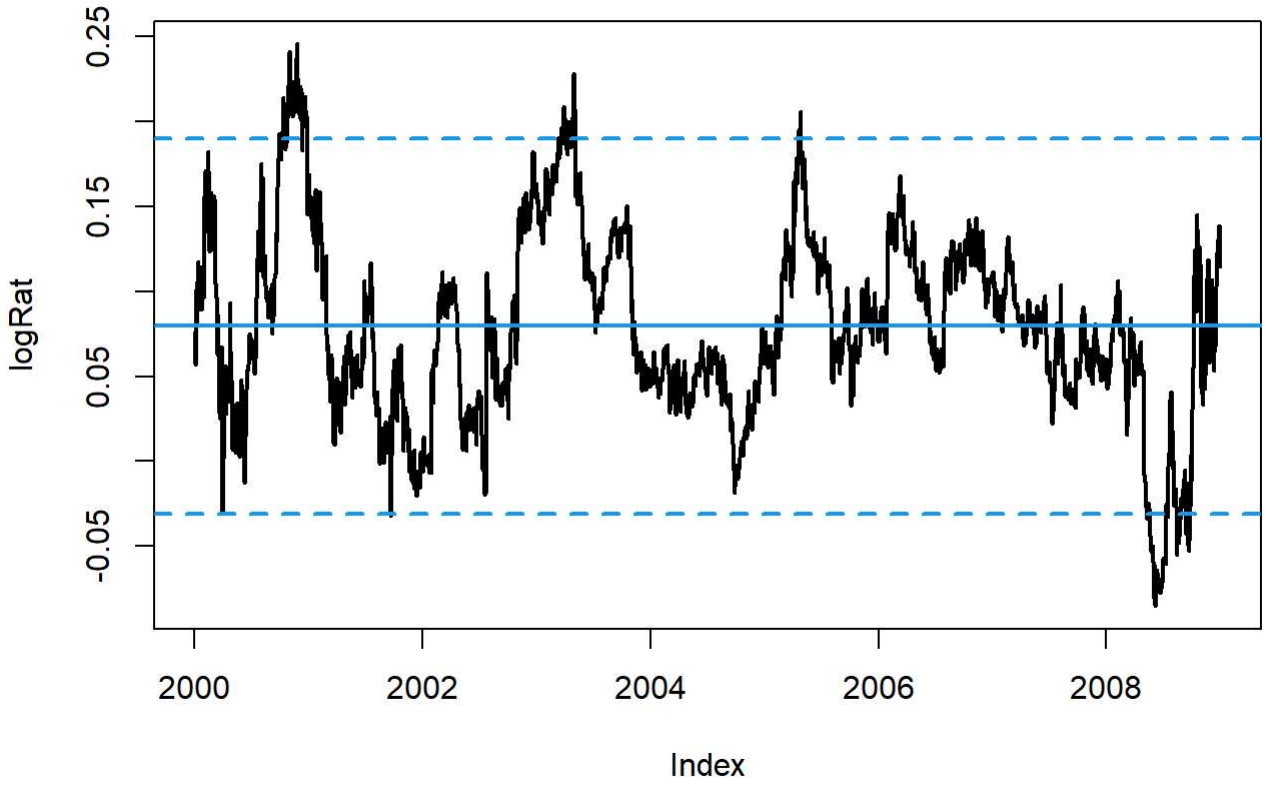
- `acf(coredata(logRat), lwd=2)` : Plots autocorrelation of log price ratio to understand temporal dependencies.
    - `coredata(logRat)` : extracts just the numerical data (the actual values), ignoring the time/index.
        - Because `acf()` works on regular numeric vectors, not all time series classes like `xts`. If `logRat` is an `xts` object, you'd need to pass its core data (the numeric values) to `acf()` —hence `coredata(logRat)`.
    - `acf(...)` : computes and plots the autocorrelation function.

1.2.Plot Log Prices and Spread

```
library(tseries);library(zoo)

#####
start.date=as.Date("2000-01-01"); end.date=as.Date("2008-12-31")

P1=get.hist.quote(instrument = "XOM", start = start.date, end=end.date, quote = "AdjClos
e", retclass = "zoo", quiet = T)
P2=get.hist.quote(instrument = "CVX", start = start.date, end=end.date, quote = "AdjClos
e", retclass = "zoo", quiet = T)
P=merge(P1,P2, all=FALSE); P1=P[,1]; P2=P[,2]
lP1=log(P1); lP2=log(P2)

#####
plot( lP1, lwd=2); # Plot log-prices from series lP1
lines( lP2 + mean(lP1-lP2), lwd=2, col=2) # Add adjusted lP2 to the plot
```

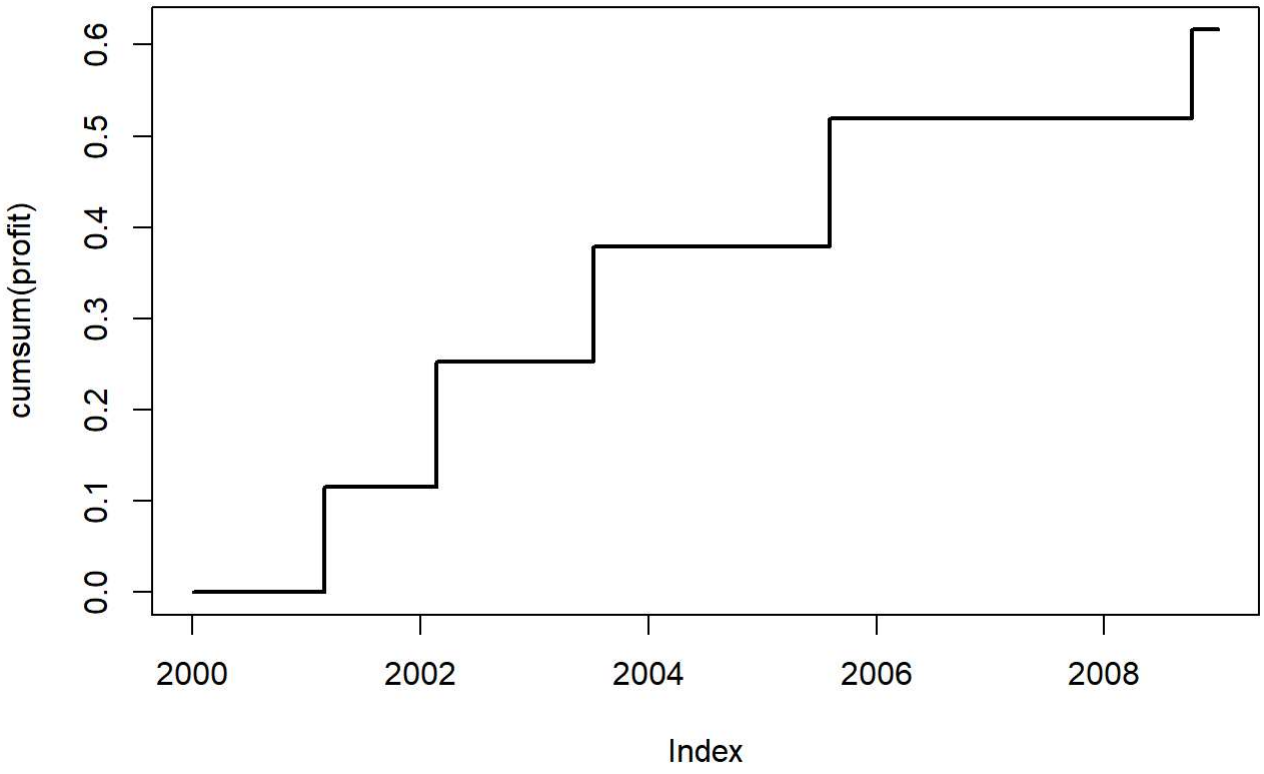



```
#### Draws ±2 standard deviation bands around the mean.
logRat= lP1-lP2 # log-ratio of prices
plot( logRat, lwd=2);
abline(h= mean(logRat), lwd=2, col=4)
abline(h=c(-2,2)*sd(logRat) + mean(logRat), col=4, lwd=2, lty=2)
```

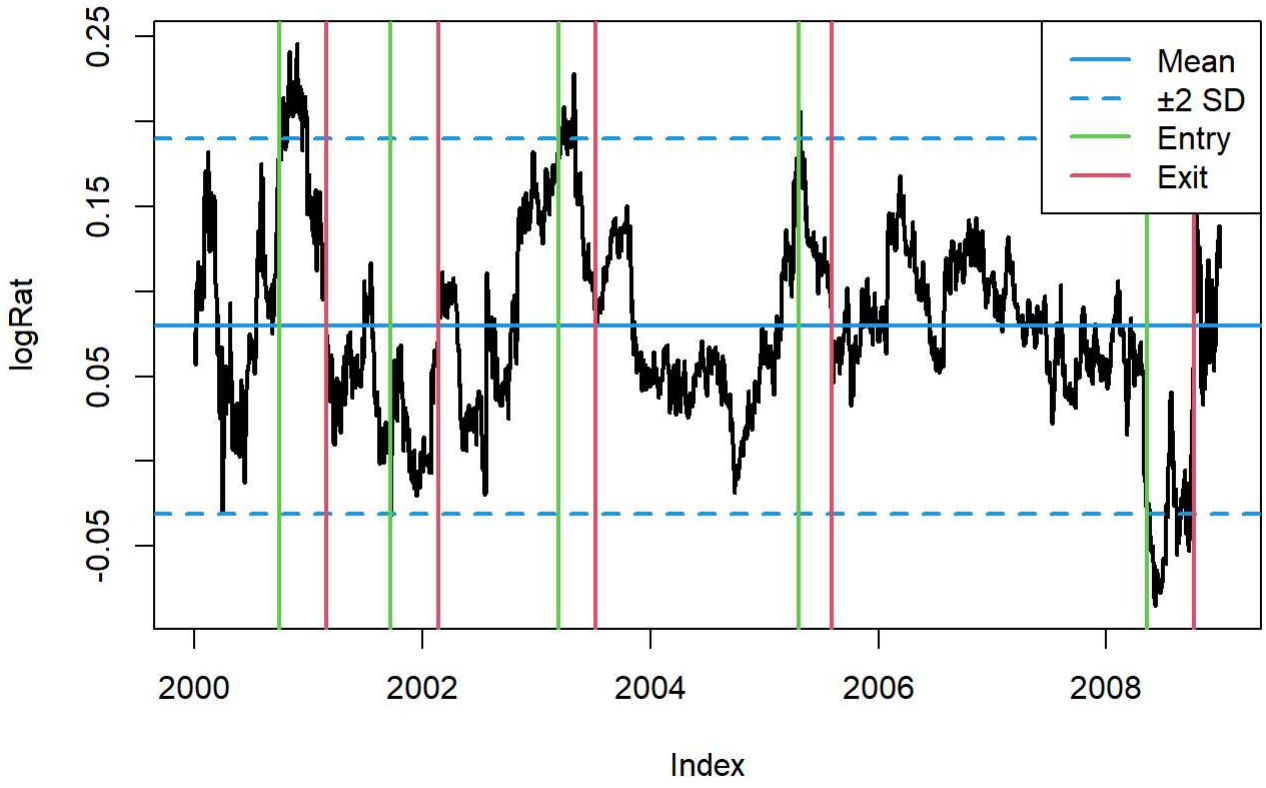

logRat
0.25
0.15
0.05
-0.05
2000
2002
2004
2006
2008
Index

1.3.Pairs Trading Logic

```
### 1.1 sets thresholds for opening/closing trades.
open.lim=2
bail.lim=3.5

### 1.2.Standardizes log-ratio;
std.logRat=(logRat-mean(logRat))/sd(logRat)

### 2.Initializes trading status and profit tracking.
trade.ind=0 # trade indicator: 0 means no position, 1 means buy 1st & sell 2nd stock, -1 m
eans sell 1st & buy 2nd stock
n=length(logRat)
profit=rep(0,n)
t.open=t.close=NULL

### 3.Main Trading Loop
for(i in 1:n){
  # Close position if mean-reversion occurred or exceed bail limit
  if( (trade.ind*std.logRat[i]<0) | (abs(std.logRat[i])>bail.lim) ){
    trade.ind=0  # No open trade
    profit[i]=profit[i]+n1*P1[i]+n2*P2[i] # Realize profit/loss
    t.close=c(t.close,i)  # Log close time
  }

  # Open position if no current trade & log-ratio deviates exceed threshold
  if( trade.ind==0 & abs(std.logRat[i])>open.lim ){
    trade.ind=sign(coredata(std.logRat[i])) # Direction of trade (+1 or -1)
    n1=-trade.ind/coredata(P1[i]) # Short or long asset 1
    n2=trade.ind/coredata(P2[i]) # Long or short asset 2
    t.open=c(t.open,i) # Log open time
  }
}

### 4.plots cumulative profit over time
profit=zoo(profit,index(P1))
plot(cumsum(profit), lwd=2)
```




```
### 5.Visualizes Spread Chart with Trade Signals

plot( logRat, lwd=2);  # log-price ratio between 2 assets (spread) as time series
abline(h= mean(logRat), lwd=2, col=4) # long-run average (blue).
abline(h=c(-2,2)*sd(logRat) + mean(logRat), col=4, lwd=2, lty=2) # standard deviation band
s (dashed blue).
abline(v=index(P1[t.open]), lwd=2, col=3) # Entry points (Green lines)
abline(v=index(P1[t.close]), lwd=2, col=2) # Exit points (Red lines)

legend("topright", legend=c("Mean", "±2 SD", "Entry", "Exit"),
       col=c(4, 4, 3, 2), lty=c(1, 2, 1, 1), lwd=2)
```

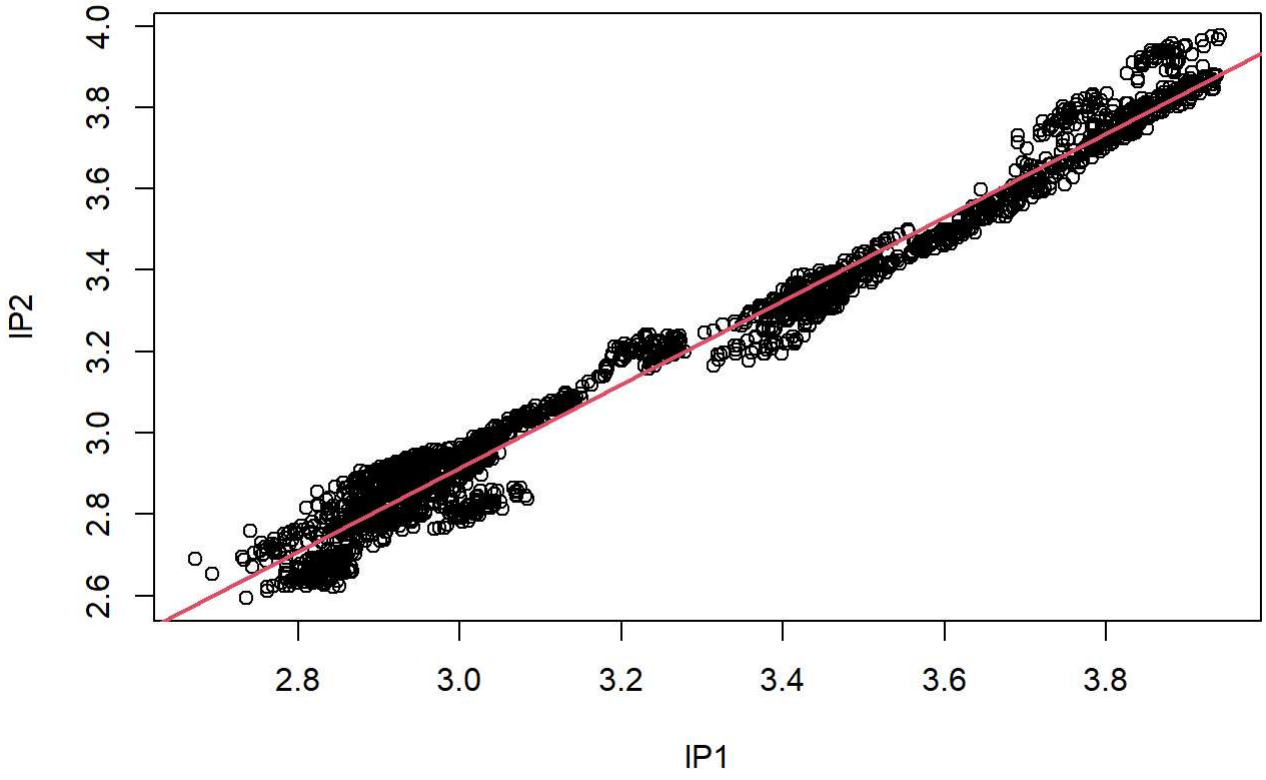


1.4.Cointegration Tests

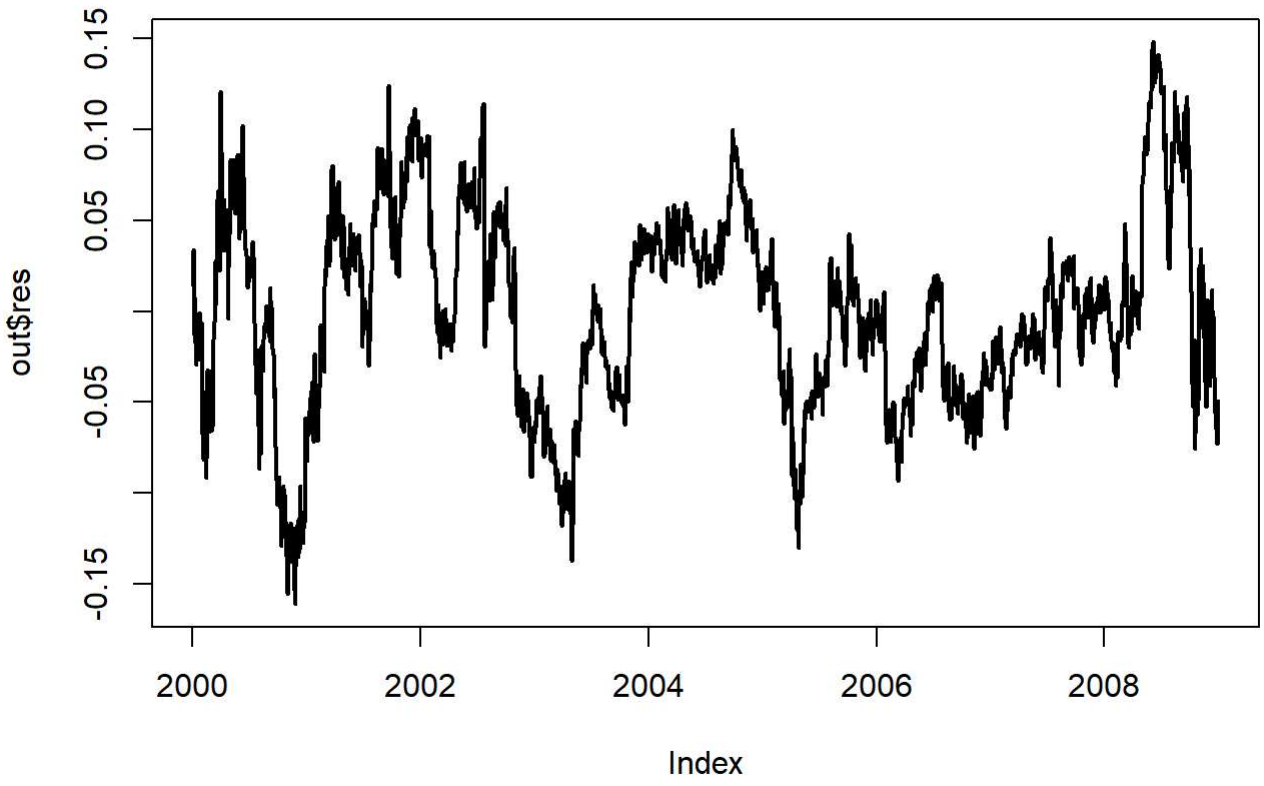
1.4.1.Two-step Engle-Granger method

- Step 1: Find the potential long-run relationship
- Step 2: Test if the difference from that relationship (residuals) is stationary.
    - Phillips–Perron test : H_0: The series has a unit root (i.e., it's non-stationary) ; H_1: The series is stationary
- If Yes => the two series are cointegrated=> usable for strategies like pairs trading.

```
#### 1.regress P2 on P1
out=lm(lP2~lP1)
plot(lP1,lP2); abline(out, lwd=2, col=2)
```




```
plot(out$res, lwd=2)
```




```
PP.test(coredata(out$res)) # Testing Residuals for Stationarity (Phillips-Perron Test)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  coredata(out$res)
## Dickey-Fuller = -4.1917, Truncation lag parameter = 8, p-value = 0.01
```

```
#### 2.regres P1 on P2 & test residuals for Stationarity
PP.test(coredata(lm(lP1~lP2)$res))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  coredata(lm(lP1 ~ lP2)$res)
## Dickey-Fuller = -4.2078, Truncation lag parameter = 8, p-value = 0.01
```

- `out = lm(lP2 ~ lP1)` : This runs a **linear regression** of the log-price of CVX (`lP2`) on the log-price of XOM (`lP1`): The fitted equation is:

$$\text{lP2} = \beta_0 + \beta_1 \cdot \text{lP1} + \epsilon$$

- where ϵ represents the residuals (the differences between the observed values and those predicted by the regression). If IP1 and IP2 are **cointegrated**, the residuals ϵ should be **stationary** (i.e., their statistical properties do not change over time). Stationary residuals indicate that any deviation from the equilibrium is temporary and mean-reverting.

1.4.2.Phillips-Ouliaris Cointegration Test Method

```
po.test(cbind(lP1,lP2)) # Phillips-Ouliaris test
```

```
## Warning in po.test(cbind(lP1, lP2)): p-value smaller than printed p-value
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  cbind(lP1, lP2)
## Phillips-Ouliaris demeaned = -36.846, Truncation lag parameter = 22,
## p-value = 0.01
```

2.Simulated Time Series and Stationarity Analysis

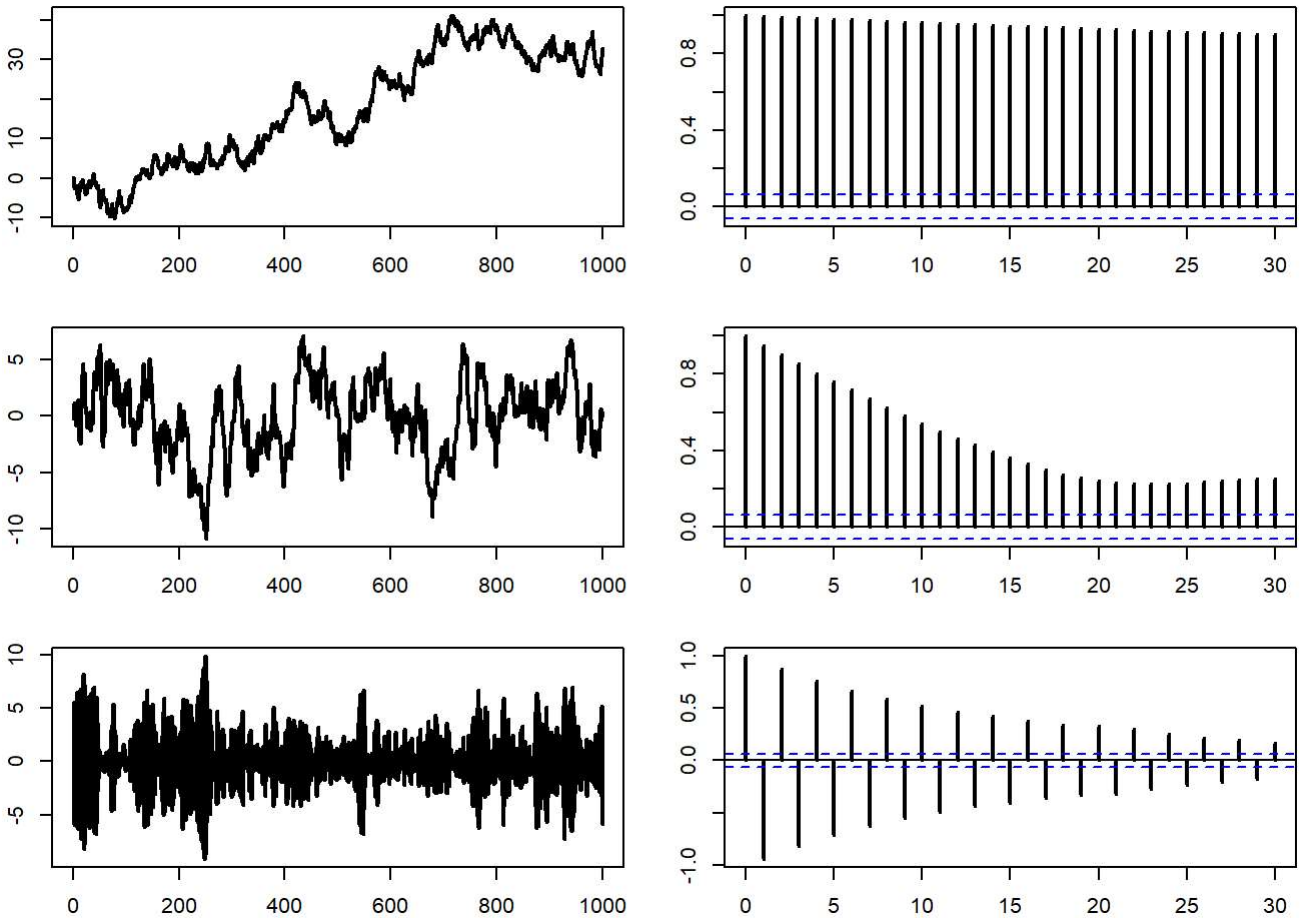
2.1.Simulated Time Series and Autocorrelations

AR(1) stands for Autoregressive model of order 1. It's a time series model where the current value depends on its immediately previous value:

$$X_t = a \cdot X_{t-1} + \varepsilon_t$$

- a is the autoregressive coefficient (also often denoted by ϕ); ε_t is white noise (random shocks at time t)
- If $a = 1$, it's a random walk (non-stationary).
- If $|a| < 1$, the process is stationary, meaning it doesn't drift off to infinity over time and has a constant mean and variance.

```
# Autocorrelations for different models

par(mfrow=c(3,2), mar=c(2,2,2,2))


### 1.Random walk (non-stationary)
X1=cumsum(rnorm(1000))
plot(X1, type='l', lwd=2, xlab="", ylab="")
acf(X1, lwd=2, main="")

### 2.stationary AR(1) with a=.95
X2=arima.sim(n=1000,list(ar=.95))
plot(X2, lwd=2)
acf(X2, lwd=2, main="")

### 3.stationary AR(1) with a=-.95
X3=arima.sim(n=1000,list(ar=-.95))
plot(X3, lwd=2)
acf(X3, lwd=2, main="")
```




```
par(mfrow=c(1,1))
```

For example an AR(1) model with coefficient $a = 0.95$, i.e.:

$$X_t = 0.95 \cdot X_{t-1} + \varepsilon_t$$

- Since the value of a is close to 1, the process exhibits strong persistence, meaning it remembers past values quite a bit.

2.2.Stationarity Tests (Phillips-Perron)

Phillips–Perron test on the time series.

H_0: The series has a unit root (i.e., it's non-stationary) ; H_1: The series is stationary

```
PP.test(X1)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  X1
## Dickey-Fuller = -2.9891, Truncation lag parameter = 7, p-value = 0.1596
```

```
PP.test(X2)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  X2
## Dickey-Fuller = -5.2906, Truncation lag parameter = 7, p-value = 0.01
```

```
PP.test(X3)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  X3
## Dickey-Fuller = -193.07, Truncation lag parameter = 7, p-value = 0.01
```

p-value > 0.05 => fail to reject the null => `X1` is likely non-stationary.

p-value < 0.05 => reject the null => `X2` , `X3` is stationary.

3.Two ETF Pairs Trading with Training Period Std Dev

Implement a pairs trading strategy of your own, using daily stock data from Yahoo Finance. More specifically, find two ETF's that are highly correlated (e.g. both track the same index or industry) and download 10 years of historical data. Use the first 5 years to select the parameters of your strategy, and the last 5 years to test it. Plot the log-difference of the series over the entire period, as well as the cumulative profit of the strategy.

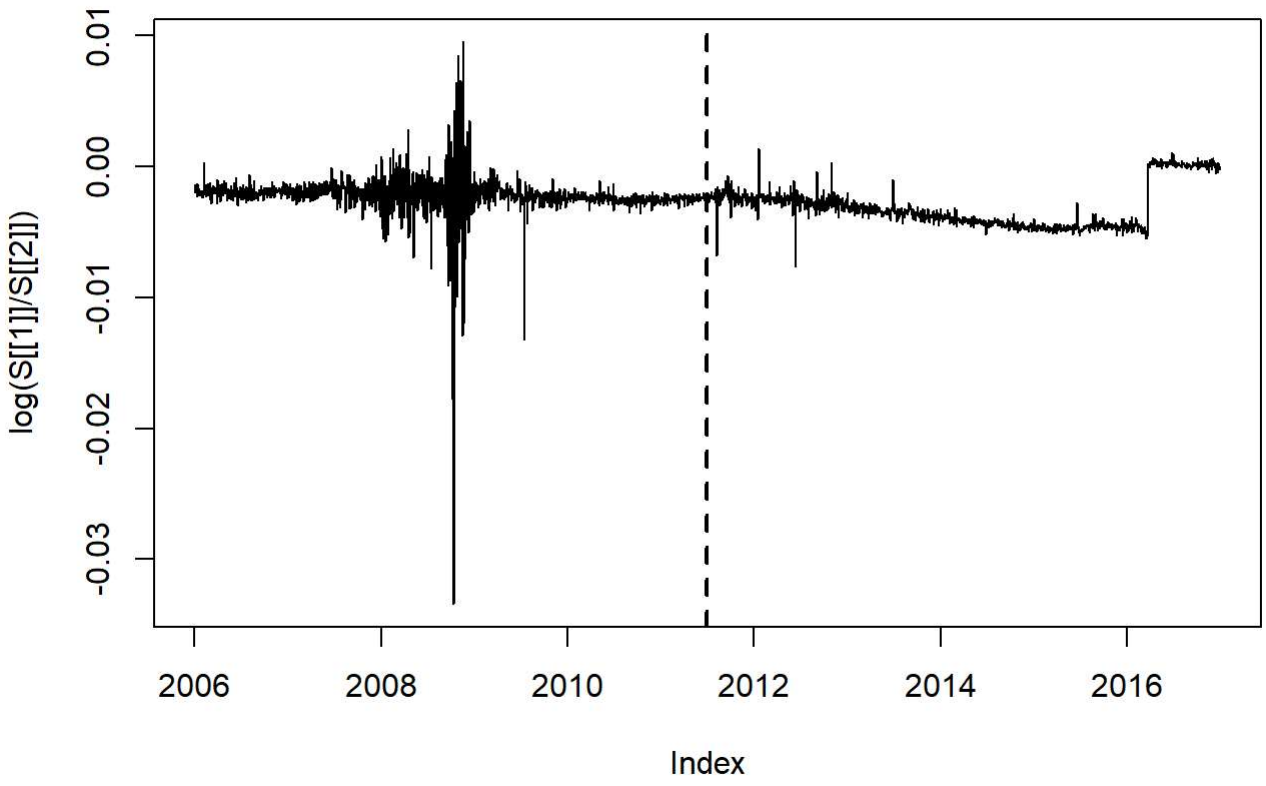
3.1.ETF’s data and Log-Ratio Plotting

We are trading “SPY” against “IVV”; both are large ETFs tracking the S&P500 index. The plot of the log-ratio of their prices is shown below:

```
library(zoo); library(tseries)

ETF.names=c('SPY','IVV')

N.ETF=length(ETF.names)
S=list() # S stores the time series for each ETF.
for(i in 1:N.ETF){
  print(paste(i, ' - ', ETF.names[i]))
  S[[i]]=get.hist.quote(ETF.names[i], start='2006-01-03', end='2016-12-30', quote='AdjClos
e')
}
```

```
## [1] "1  -  SPY"
## time series ends   2016-12-29
## [1] "2  -  IVV"
## time series ends   2016-12-29
```

```
names(S)=ETF.names

# Split data in half: first half for training, second half for testing.
n=length(S[[1]]); n.half=n%/%2

# Plot log-ratio of prices: log(S1/S2).
plot(log(S[[1]]/S[[2]]), type='l'); abline(v=index(S[[1]])[n.half], lty=2, lwd=2)
```



Note that the log-ratio exhibits highly mean reverting behavior in the training period (left half), especially around the Fall 2008 financial crisis. In the testing period (right half), there are less intense fluctuations, and an adjustment (jump) at the beginning of 2016.

3.2.Backtesting Pairs Trading Using Standardized Log-Price Ratio

Use the first 5 years to select the parameters of your strategy, and the last 5 years to test it. Plot the log-difference of the series over the entire period, as well as the cumulative profit of the strategy.

The pairs trading strategy employed involves opening a position at 2 standard deviations, closing a position at the mean, and bailing-out a position at 4 standard deviations (where the **standard deviation is calculated based on the first half of data**). The resulting strategy and cumulative gain (per $1 position in either stock) are shown below:

```
## INPUT
open.lim=2
bail.lim=4


### 1.Train Test Split -> std log-price ratio

S.1.train=coredata(S[[1]][1:n.half])
S.2.train=coredata(S[[2]][1:n.half])

S.1.test=coredata(S[[1]][(n.half+1):n])
S.2.test=coredata(S[[2]][(n.half+1):n])

profit.PP=rep(0,length(S.1.test))

logRatio.train=log(S.1.train/S.2.train)
logRatio.test=log(S.1.test/S.2.test)

std.logRatio.test=(logRatio.test-mean(logRatio.train))/sd(logRatio.train)

### 2.Pairs Trading
trade.ind=0 # trade indicator: 0 means no position, -1 means buy 1st & sell 2nd stock, 1 m
eans sell 1st & buy 2nd stock
bail.ind=0 # bail-out indicator

t.open=t.close=NULL
for(j in 1:length(logRatio.test)){
  # check for closing an open position
  if( (trade.ind*std.logRatio.test[j]<0 | abs(std.logRatio.test[j])>bail.lim) & bail.ind==
0 ){
    trade.ind=0
    profit.PP[j]=profit.PP[j]+n1*S.1.test[j]+n2*S.2.test[j]
    t.close=c(t.close,j)
    if( abs(std.logRatio.test[j])>bail.lim ){
      bail.ind=1 # bail out position, & stop trading
    }
  }
  # check for opening a position
  if( (trade.ind==0) & (abs(std.logRatio.test[j])>open.lim) & (bail.ind==0) ){
    trade.ind=sign(std.logRatio.test[j])
    n1=-trade.ind/S.1.test[j]
    n2=trade.ind/S.2.test[j]
    t.open=c(t.open,j)
  }
}


### 3.Plotting Results
par(mfrow=c(1,2))
plot(cumsum(profit.PP), type='l', ylab="cumulative profit per $1 invested", lwd=2)

plot(std.logRatio.test, type='l', lwd=2, ylim=c(-4,4))
```

```
abline(h=c(-2,2), col=5); abline(h=c(-4,0,4), col=4)
abline(v=t.open, col=3, lty=2); abline(v=t.close, col=2, lty=2)
```