Quantitative Finance Assignment

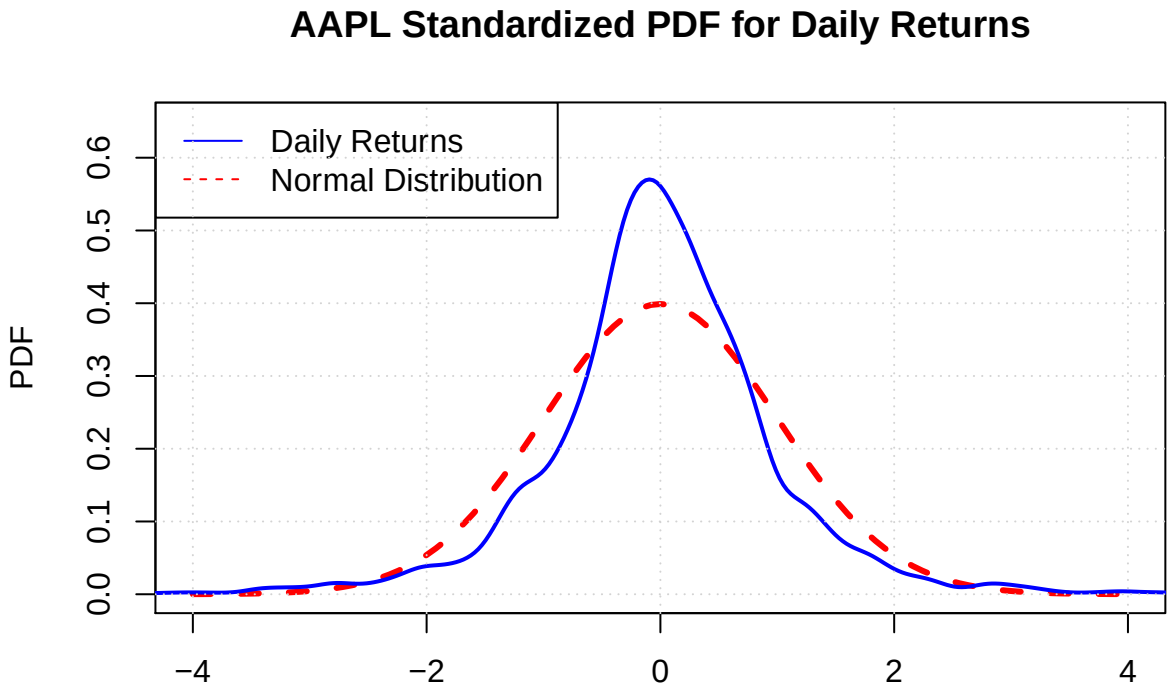
Ning Zhao

March 4, 2017

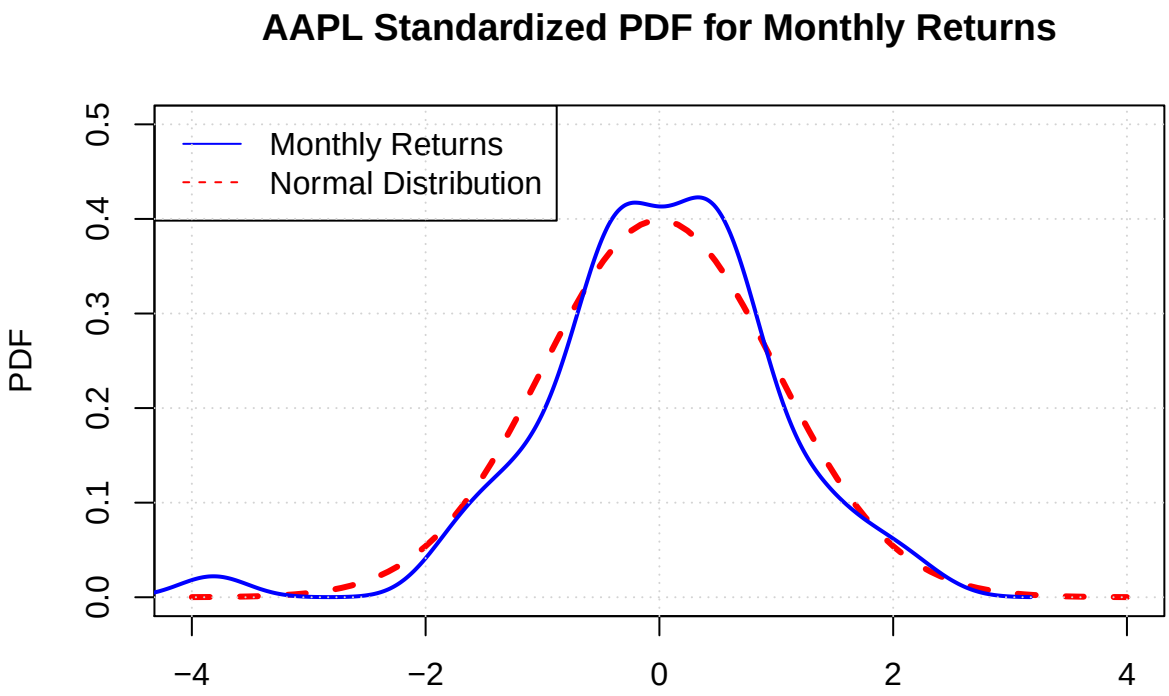
1.How good is the assumption of normal distributions for financial returns?

We say it is good because this assumption helps to create a simple financial theoretical background. Like the CAPM and BS formula, they both have assumption that the returns are normally distributed. Therefore, although this assumption is unrealistic, it facilitates a lot of statistical tools and pushes academics. Actually, the distributions for financial returns depend on the time horizon. For short-term returns, the distribution is semi-heaviness and has non-zero skewness, large kurtosis. For the long-term returns, the distribution looks more normal.

```
library(quantmod)
data <- getSymbols("AAPL", auto.assign = F)
# calculate daily returns and monthly returns
data_rtn_d <- dailyReturn(data$AAPL.Adjusted)
data_rtn_m <- monthlyReturn(data$AAPL.Adjusted)
# standardization
data_rtn_d_norm <- (data_rtn_d-mean(data_rtn_d))/sd(data_rtn_d)
data_rtn_m_norm <- (data_rtn_m-mean(data_rtn_m))/sd(data_rtn_m)
# plot probability density function for standardized daily
# returns and standard normal distribution
plot(dnorm,-4,4,ylim=c(0,0.65),xlab="",ylab="PDF",col=2,cex=9,
     lty=2,lwd=3,main="AAPL Standardized PDF for Daily Returns")
lines(density(data_rtn_d_norm),xlim=c(-4,4),col=4,lwd=2)
legend("topleft",c("Daily Returns","Normal Distribution"),lty=c(1,2),col=c(4,2))
grid()
```




```
# plot pdf for standardized monthly returns and standard normal distribution
plot(dnorm,-4,4,ylim=c(0,0.5),xlab="",ylab="PDF",col=2,cex=9,
     lty=2,lwd=3,main="AAPL Standardized PDF for Monthly Returns")
lines(density(data_rtn_m_norm),xlim=c(-4,4),col=4,lwd=2)
legend("topleft",c("Monthly Returns","Normal Distribution"),lty=c(1,2),col=c(4,2))
grid()
```



2.Simulate options pricing using the binomial model. Why would you use this model instead of the Black-and-Scholes model?

BS model provides an analytical formula to get the risk neutral price. But it doesn't offer the flexibility required to value options with non-standard features, such as a price reset feature or a mandatory exercise requirement. Binomial method provides a numerical procedure to price option. The most obvious advantage of binomial model is path-dependent option pricing. It can be used to accurately price American options.

1) Binomial Tree of European Option

Multiplicative Binomial Tree of European Option

EX. One-year maturity, at-the-money European call option with current asset price at 100. The binomial tree has 10 time steps and up and down proportional jumps of 1.1 and 0.9091 respectively. The continuously compounded interest rate is 6 percent per annum.

The option price is:

```
Binomial_E <- function(type, K, T, S, r, N, u, d)
{
  dt <- T / N
  p <- (exp(r*dt)-d)/(u-d)
  disc <- exp(-r*dt)
  # get stock prices
  s <- c()
```

```
  s[1] <- S*u^N
  for (i in 2:(N+1))
  {
    s[i] <- s[i-1]*d/u
  }
  # initialise option values at maturity
  if(type=="Call")
  {
    M <- as.matrix(pmax(s-K,0))
  } else{
    M <- as.matrix(pmax(K-s,0))
  }
  # use matrix to calculate back the call option value
  for (i in 1:N)
  {
    # delete the last line in matrix M
    M1 <- M[-nrow(M),]
    # delete the first line in matrix M
    M2 <- M[-1,]
    M <- as.matrix(disc*(p*M1+(1-p)*M2))
    # print(M)
  }
  M[1,1]
}
Binomial_E("Call", 100, 1, 100, 0.06, 10, 1.1, 1/1.1)
```

```
## [1] 14.47754
```

General Additive Binomial Valuation of European Option

EX. One-year maturity, at-the-money European call option with current asset price at 100 and volatility of 20 percent. The binomial tree has 10 time steps. The continuously compounded interest rate is 6 percent per annum.

```
Binomial_Euro <- function(type, K, T, S, r, N, sig, div)
{
  dt <- T / N
  nu <- r-div-0.5*sig^2
  dxu <- sqrt(sig^2*dt+(nu*dt)^2)
  dxd <- -dxu
  pu <- 0.5+0.5*(nu*dt/dxu)
  pd <- 1-pu
  disc <- exp(-r*dt)
  # get stock prices
  s <- c()
  s[1] <- S*exp(N*dxu)
  for (i in 2:(N+1))
  {
    s[i] <- s[i-1]*exp(dxd-dxu)
  }
  # initialise option values at maturity
  if(type=="Call")
  {
```

```
    M <- as.matrix(pmax(s-K,0))
  } else{
    M <- as.matrix(pmax(K-s,0))
  }
  # use matrix to calculate back the call option value
  for (i in 1:N)
  {
    # delete the last line in matrix M
    M1 <- M[-nrow(M),]
    # delete the first line in matrix M
    M2 <- M[-1,]
    M <- as.matrix(disc*(pu*M1+pd*M2))
    # print(M)
  }
  M[1,1]
}
Binomial_Euro("Call", 100, 1, 100, 0.06, 10, 0.2, 0)
```

```
## [1] 10.80106
```

2) Binomial Tree of Amercian Option

Multiplicative Binomial Tree of Amercian Option

EX. One-year maturity, at-the-money Amercian put option with current asset price at 100. The binomial tree has 10 time steps and up and down proportional jumps of 1.1 and 0.9091 respectively. The continuously compounded interest rate is 6 percent per annum.

```
Binomial_A <- function(type, K, T, S, r, N, u, d)
{
  dt <- T / N
  p <- (exp(r*dt)-d)/(u-d)
  disc <- exp(-r*dt)
  # get stock prices
  s <- c()
  s[1] <- S*u^N
  for (i in 2:(N+1))
  {
    s[i] <- s[i-1]*d/u
  }
  # initialise option values at maturity
  if(type=="Call")
  {
    M <- as.matrix(pmax(s-K,0))
  } else{
    M <- as.matrix(pmax(K-s,0))
  }
  for (i in 1:N)
  {
    # delete the last line in matrix M
    M1 <- M[-nrow(M),]
    # delete the first line in matrix M
    M2 <- M[-1,]
```

```
    M <- disc*(p*M1+(1-p)*M2)
    s <- as.matrix(s)[-1,]/d
    # judge it worth to execute before expiration or not
    if(type=="Call")
    {
      M <- as.matrix(pmax(M, s-K))
    } else{
      M <- as.matrix(pmax(M, K-s))
    }
    # print(M)
  }
  M[1,1]
}
Binomial_A("Put", 100, 1, 100, 0.06, 10, 1.1, 1/1.1)
```

```
## [1] 9.450911
```

General Additive Binomial Valuation of Amercian

EX. One-year maturity, at-the-money Amercian put option with current asset price at 100 and volatility of 20 percent. The binomial tree has 10 time steps. The continuously compounded interest rate is 6 percent per annum.

```
Binomial_Ameri <- function(type, K, T, S, r, N, sigma, div)
{
  dt <- T/N
  nu <- r-div-0.5*sigma^2
  dxu <- sqrt(sigma^2*dt+(nu*dt)^2)
  dxd <- -dxu
  pu <- 0.5+0.5*(nu*dt/dxu)
  pd <- 1-pu
  disc <- exp(-r*dt)
  dpu <- disc*pu
  dpd <- disc*pd
  edxdu <- exp(dxd-dxu)
  edxd <- exp(dxd)
  s <- c()
  s[1] <- S*exp(N*dxu)
  for (i in 2:(N+1))
  {
    s[i] <- s[i-1]*edxdu
  }
  if(type=="Call")
  {
    M <- as.matrix(pmax(s-K,0))
  } else{
    M <- as.matrix(pmax(K-s,0))
  }
  for (i in 1:N)
  {
    M1 <- M[-nrow(M),]
    M2 <- M[-1,]
    M <- dpu*M1+dpd*M2
```

```
    s <- as.matrix(s)[-1,]/edxd
    # judge it worth to execute before expiration or not
    if(type=="Call")
    {
      M <- as.matrix(pmax(M, s-K))
    } else{
      M <- as.matrix(pmax(M, K-s))
    }
    #print(M)
  }
  M[1,1]
}
Binomial_Ameri("Put", 100, 1, 100, 0.06, 3, 0.2, 0)
```

```
## [1] 6.162109
```

3) Additive Binomial Valuation of Amercian with a Discrete Proportional Dividend

EX. One-year maturity, at-the-money Amercian put option with current asset price at 100 and volatility of 20 percent. The binomial tree has 3 time steps. The continuously compounded interest rate is 6 percent per annum. The asset is assumed to pay a discrete proportional dividend of 3 percent after eight month.

```
Binomial_Ameri_Proportion <- function(type, K, T, S, r, N, sigma, div, dvh, tau)
{
  dt <- T/N
  nu <- r-div-0.5*sigma^2
  dxu <- sqrt(sigma^2*dt+(nu*dt)^2)
  dxd <- -dxu
  pu <- 0.5+0.5*(nu*dt/dxu)
  pd <- 1-pu
  disc <- exp(-r*dt)
  dpu <- disc*pu
  dpd <- disc*pd
  edxdu <- exp(dxd-dxu)
  edxd <- exp(dxd)
  # calculate # of the dividen
  n <- floor((T-tau)/(T/N))+1
  s <- c()
  s[1] <- S*exp(N*dxu)*(1-dvh)
  for (i in 2:(N+1))
  {
    s[i] <- s[i-1]*edxdu
  }
  # print(s)
  if(type=="Call")
  {
    M <- as.matrix(pmax(s-K,0))
  } else{
    M <- as.matrix(pmax(K-s,0))
  }
  for (i in 1:N)
  {
```

```
    if(i>=N-n+1)
    {
      s <- s/(1-dvh)
    }
    M1 <- M[-nrow(M),]
    M2 <- M[-1,]
    M <- dpu*M1+dpd*M2
    s <- as.matrix(s)[-1,]/edxd
    # judge it worth to execute before expiration or not
    if(type=="Call")
    {
      M <- as.matrix(pmax(M, s-K))
    } else{
      M <- as.matrix(pmax(M, K-s))
    }
    #print(M)
  }
  M[1,1]
}
Binomial_Ameri_Proportion("Put", 100, 1, 100, 0.06, 3, 0.2, 0, 0.03, 2/3)
```

```
## [1] 7.159079
```

3.Estimate Equity Portfolio Risk without using asset prices.

Naturally, risk is dependent on the market environment and investor sentiment. Without using asset prices, the implied volatility of corresponding options can be used to do principal component analysis. Demo code:

```
# get the implied vol of options
implvol
# the correlation matrix
R <- cor(impvol)
# the eigenvalues of the correlation matrix, choose a sufficient number of factors
eigenR <- eigen(R)
plot(eigenR$values,type='o',xlab='Component',ylab="Eigenvalues")

# Assume the sufficient number of factors is 2
#use the PCA method to find a final rotated factor solution
pc <- principal(implvol, nfactors=2, rotate ="varimax")
loadings(pc)
#build a factor model to predict
factor <- implvol %*% pc$loadings
f1 <- factor[,1]
f2 <- factor[,2]
lm <- lm(implvol~f1+f2)
factor_model <- lm$coefficients[1]+lm$coefficients[2]*f1+lm$coefficients[3]*f2
```

The problem here is there are many kinds of options have the same underlying security but with different maturities and strike prices. So it's hard to choose which implied volatility is suitable to use. Furthermore, we can use quantified news to predict risk. We need to transform raw news to quantitative inputs and use the information about the changing market sentiment to update our beliefs about factor volatility.

4.Use freely available data from the web to predict/explain macroeconomic indicators. Financial/Economic/Fundamentals data are not allowed.

We can use text data from the web like Twitter or Google Trends to predict macroeconomic indicators.

EX. US Unemployment Rate Prediction

Selecting Features: use daily job search activity as feature.

When people seeking work, they always use terms like: jobs, resume, employment, indeed.com. Daily job search activity is the sum of the number of searches performed using these four job-search terms.

Demo code:

```
# Python Implementation
from sklearn import svm
# get the daily job search activity for the last year
jobsearch_train = []
# get the unemployment rate for the last year
rate_train = []
# implement SVM
clf = svm.SVR(kernel='rbf',C=  ,gamma=  )
# get the daily job search activity for the recent 10 days
jobsearch_test = []
# prediction
rate_test = clf.fit(jobsearch_train, rate_train).predict(jobsearch_test)
```

5.Implement one Smart Beta strategy and discuss pros and cons compared to a chosen benchmark.

Smart beta strategies: moving away from traditional market capitalisation-based indices to alternative strategies, in search of better returns and lower costs amid volatile markets and an uncertain economic climate.

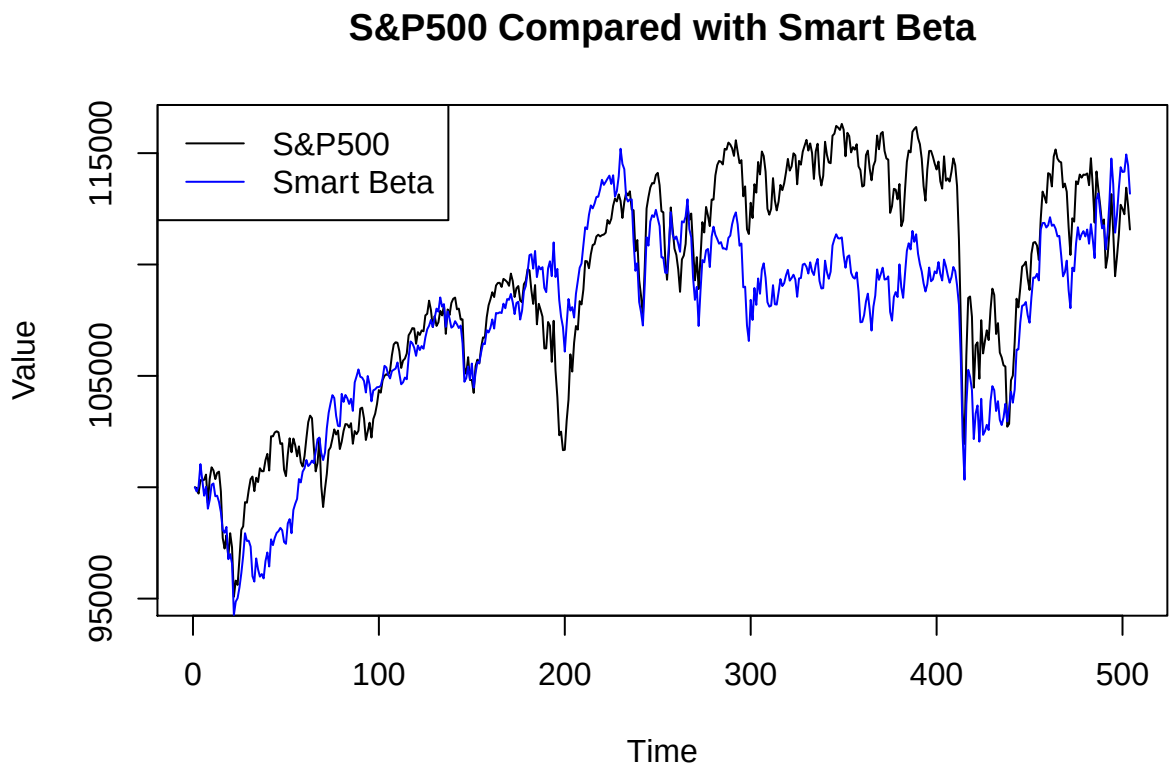
Strategy EX. risk parity

$$w_i = \frac{1/\sigma_i}{\sum 1/\sigma_j}$$

I choose S&P500 as benchmark and calculate the volatility of returns for 30 largest market capitalization stocks in the S&P500 index. I download data from 2014/01/01 to 2017/01/01 and use the first two-year data as training data, the recent year as testing data. I select stocks whose volatility of returns are less than 0.01 to construct portfolio. Setting the orginal captical as 100000, the performances are,

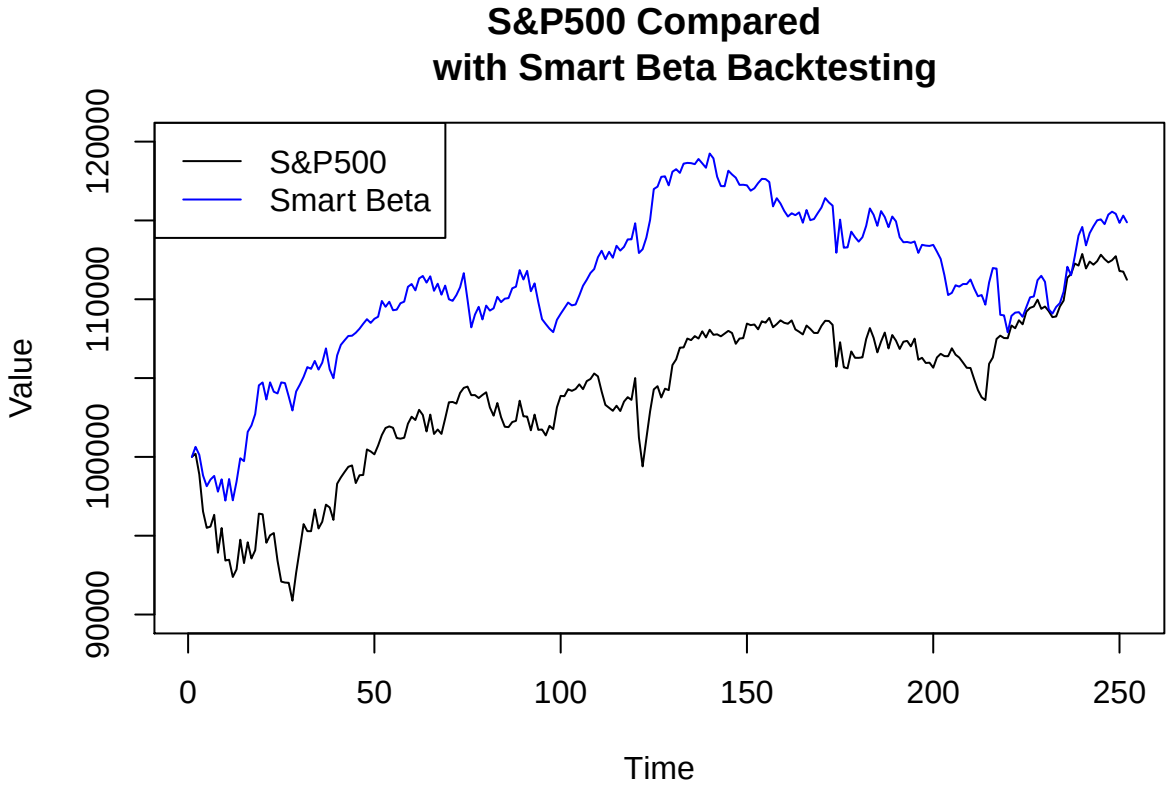
```
# choose SP500 as benchmark
startdate <- "2014-01-01"
enddate <- "2017-01-01"
sp500 <- getSymbols("^GSPC",from=startdate,to=enddate, auto.assign = F)
combine <- sp500[,6]
# find 30 stocks with the largest weights in SP500
name <- c("AAPL","XOM","JNJ","PG","IBM","JPM","T","GE","CVX",
              "PFE","WFC","CSCO","KO","BAC","HPQ","WMT","INTC","MRK","PEP",
              "ORCL","VZ","PM","GS","ABT","SLB","QCOM","COP","C","MCD","OXY")
# combine adjusted prices of 30 stocks together
for(i in 1:length(name))
{
  price <- getSymbols(name[i],from=startdate,to=enddate, auto.assign = F)[,6]
```

```
  combine <- merge(combine,price)
}
# two years for training, one year for test
train <- combine[1:(nrow(combine)-252),]
test <- tail(combine,252)
# calculate log returns
return <- diff(log(train))[-1,]
vol <- apply(return,2,sd)
# choose volatility less than 0.01 to construct portfolio
colnum <- which(vol<0.01)
portfolio <- train[,colnum]
# weight for each stock
w <- 1/vol[colnum][-1]/sum(1/vol[colnum][-1])
# performance
capital <- 100000
share <- capital*c(1,w)/portfolio[1,]
v_b <- as.numeric(share[1,1])*portfolio[,1]
v_p <- portfolio[,-1] %*% t(share[,-1])
# plot
plot(ts(v_b),ylab="Value",main="S&P500 Compared with Smart Beta")
lines(ts(v_p),col=4)
legend("topleft",c("S&P500","Smart Beta"),lty=c(1,1),col=c(1,4))
```




```
# out of sample back testing
portfolio_t <- test[,colnum]
```

```r
capital <- 100000
share <- capital*c(1,w)/portfolio_t[1,]
v_b <- as.numeric(share[1,1])*portfolio_t[,1]
v_p <- portfolio_t[,-1] %*% t(share[,-1])
# plot
plot(ts(v_b),ylab="Value",ylim=c(90000,120000),main="S&P500 Compared
     with Smart Beta Backtesting")
lines(ts(v_p),col=4)
legend("topleft",c("S&P500","Smart Beta"),lty=c(1,1),col=c(1,4))
```



I want to redefine an index with lower volatility than benchmark. But the volatility of returns for S&P500 is already small. I just calculate volatality for 30 stocks so it's hard to select suitable stocks. Therefore, through a long period, the performance of the benchmark and this smart Beta strategy should be similar.

6.

a) Suggest one data source that might be useful to explain or predict the FX market.

b) Derive and discuss relevant analytics from this data source.

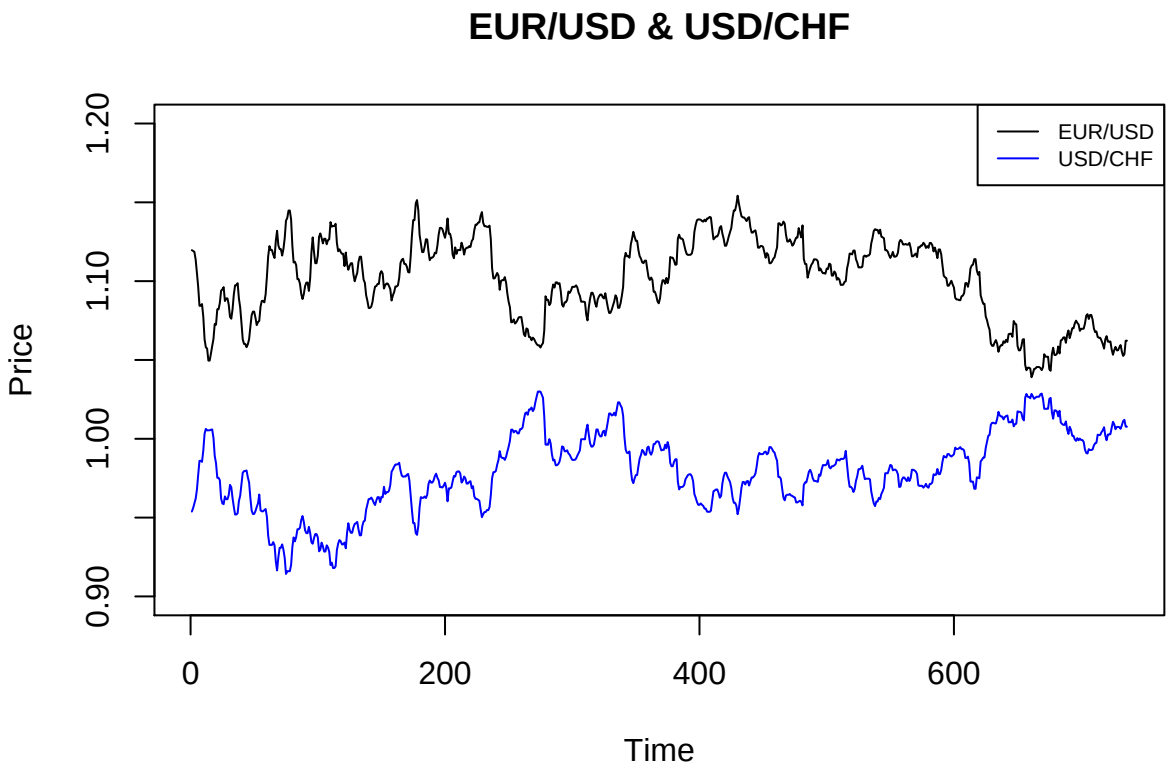
c) Determine whether your proposed analytics are co-integrated with currency pairs.

d) Describe and implement a pairs trading strategy exploiting your analytics.

DAILYFX: It provides News including forcast, market alert and Technical Analysis including support&resistance, daily technical reports.

Pairs Trading Strategy:

Using EUR/USD and USD/CHF, they seem like co-integrated currency pairs



Strategy:

calculate the price difference seperatly and get the hedge ratio by linear regression;

get the spread price, then calculate the mean and standard deviation of spread price;

set the bound: upper=mean+sd; lower=mean-sd;

when the spread exceed the upper bound, sell EUR/USD and buy USD/CHF; when the spread is smaller than the lower bound, sell USD/CHF and buy EUR/USD

```r
# define training set
startT  <- "2015-03-01"
endT    <- "2016-03-01"
rangeT  <- paste(startT,"::",endT,sep ="")
EURUSD_T   <- EURUSD[rangeT]
USDCHF_T   <- USDCHF[rangeT]
#define out of sample set
startO  <- "2016-03-01"
endO    <- "2017-03-01"
rangeO  <- paste(startO,"::",endO,sep ="")
EURUSD_O   <- EURUSD[rangeO]
USDCHF_O   <- USDCHF[rangeO]

# compute price difference
pd_eu <- diff(EURUSD_T)[-1]
pd_uc <- diff(USDCHF_T)[-1]
model <- lm(pd_eu ~ pd_uc)
# get the hedge ratio
hedgeratio <- as.numeric(model$coefficients[2])
# spread price
spread <- EURUSD_T-hedgeratio*USDCHF_T
mean_sp <- mean(spread)
sd_sp <- sd(spread)
# set lower and upper bound
upper <- mean_sp+sd_sp
lower <- mean_sp-sd_sp
# when the spread exceed the upper bound, sell EUR/USD and buy USD/CHF
# when the spread is smaller than the lower bound, sell USD/CHF and buy EUR/USD
sell <- which(spread>=upper)
buy <- which(spread<=lower)
```