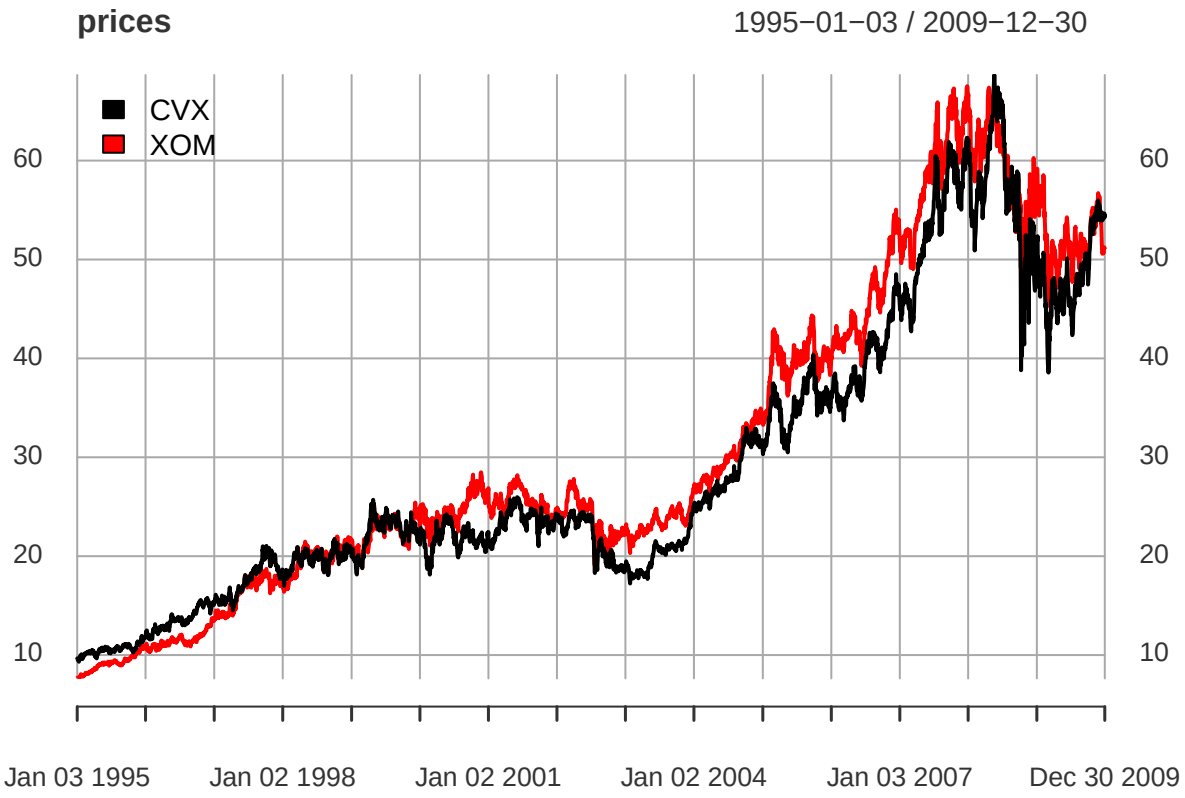
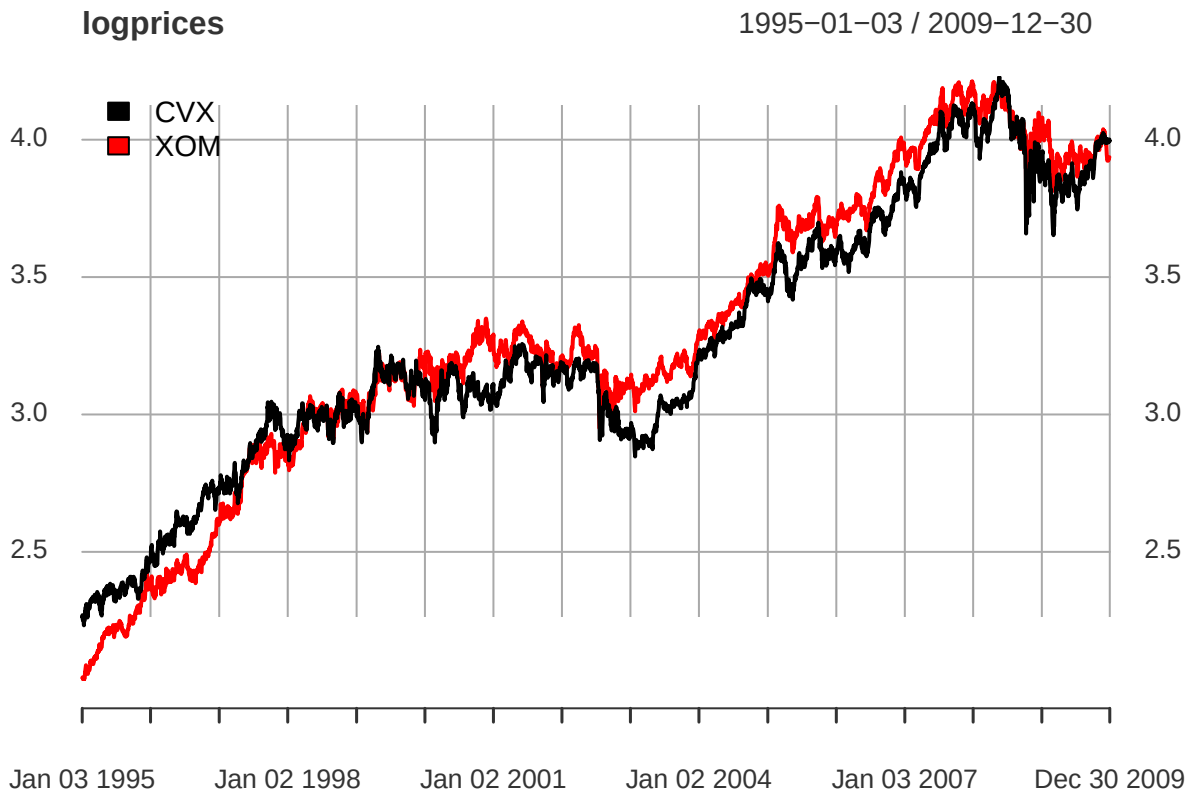
Pair trading improved version 2

Data Input

```
library(quantmod)
library(xts)
begin_date <- "1995-01-02"
end_date <- "2009-12-31"
stock_namelist <- c("CVX", "XOM")
prices <- xts()
for (stock in 1:length(stock_namelist))
  prices <- cbind(prices, Ad(getSymbols(stock_namelist[stock],
                             from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(prices) <- stock_namelist
indexClass(prices) <- "Date"
plot(prices, legend.loc = "topleft")
```




```
logprices <- log(prices)
plot(logprices, legend.loc = "topleft")
```



Linear Regression

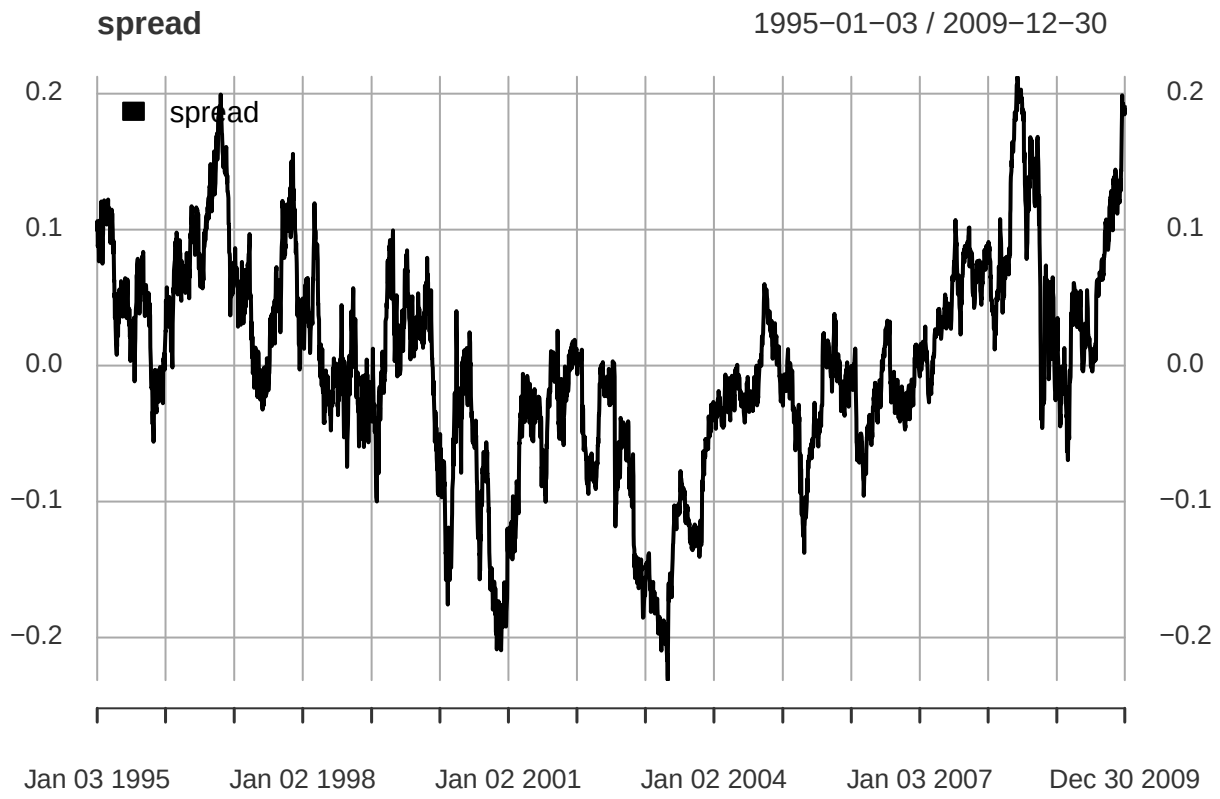
```
T <- nrow(prices)
frac <- 1 # if we use the whole sample as our training set
T_trn <- round(frac*T)
T_tst <- T - T_trn
y1 <- logprices[,1]
y2 <- logprices[,2]
ls_coeffs <- coef(lm(y1[1:T_trn] ~ y2[1:T_trn]))
ls_coeffs
```

```
## (Intercept) y2[1:T_trn]
##   0.3870422   0.8700739
```

```
mu <- ls_coeffs[1]
gamma <- ls_coeffs[2]
```

plot the spread to observe its mean-reversion property
The spread is defined as: $z_t = y_{1t} - \gamma y_{2t} - \mu$

```
spread <- y1 - gamma*y2 - mu
colnames(spread) <- "spread"
plot(spread, legend.loc = "topleft")
```

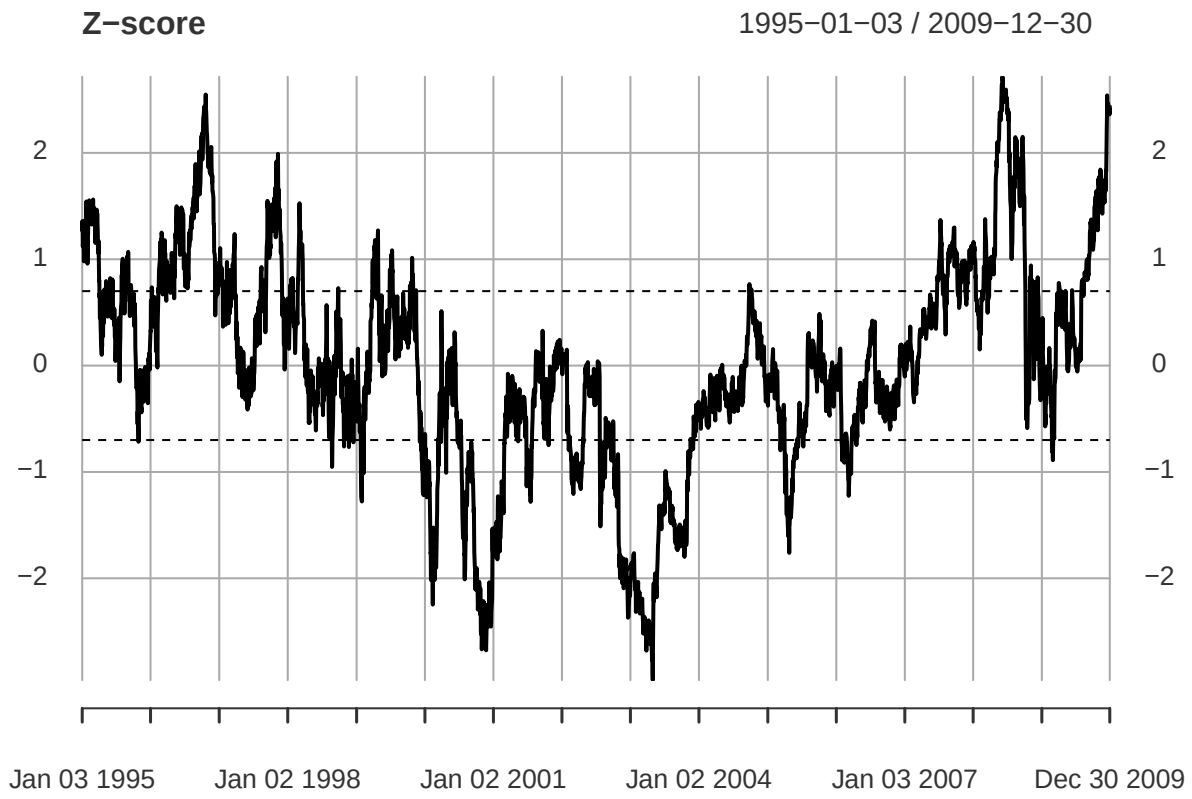


UNSOLVED PROBLEM
whether the cointegration is persistent

Trading signal: z-score

$$Z_t^{score} = \frac{z_t - E[z_t]}{Std[z_t]}$$

```
spread_mean <- mean(spread)
spread_volatility <- sd(spread)
z_score <- (spread - spread_mean)/spread_volatility
colnames(z_score) <- "z_score"
upper_bound <- xts(rep(0.7, T_trn), index(z_score))
lower_bound <- xts(rep(-0.7, T_trn), index(z_score))
{ plot(z_score, main = "Z-score")
  lines(upper_bound, lty = 2)
  lines(lower_bound, lty =2)}
```



Define a function for identifying trading signal: 0 for no position, 1 for long spread, -1 for short spread

```
generate_signal <- function(z_score, upper_bound, lower_bound){
  #set initial position
  signal <- z_score
  colnames(signal) <- "signal"
  signal[] <- NA
  if (z_score[1] > upper_bound){
    signal[1] <- -1
  } else if (z_score[1] < lower_bound){
    signal[1] <- 1
  } else {signal[1] <- 0}

  for (t in 2:nrow(z_score)) {
    if (signal[t-1] == 0) {
      if (z_score[t] >= upper_bound) signal[t] <- -1
      else if (z_score[t] <= lower_bound) signal[t] <- 1
      else signal[t] <- 0
    }else if (signal[t-1] == 1){
      if (z_score[t] >= 0) signal[t] <- 0
      else signal[t] <- 1
    }else {
      if (z_score[t] <= 0) signal[t] <- 0
      else signal[t] <- -1
    }
  }
  return(signal)
```

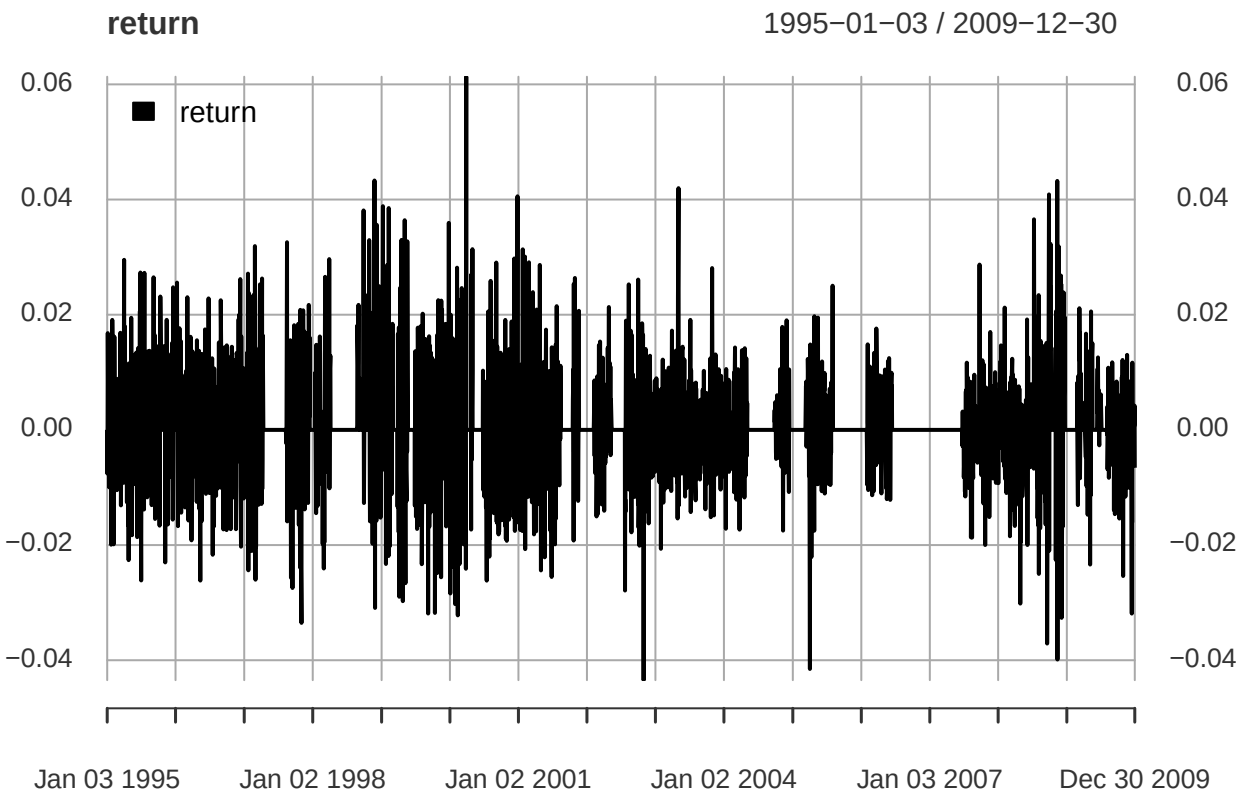
```
}
signal <- generate_signal(z_score, 0.7, -0.7)
{plot(cbind(z_score, signal), legend.loc = "topleft", main = "Implement of pair trading strategy over ti
  lines(upper_bound, lty = 2)
  lines(lower_bound, lty = 2)}
```

Compute the P&L of the strategy over time.

Two methods for P&L

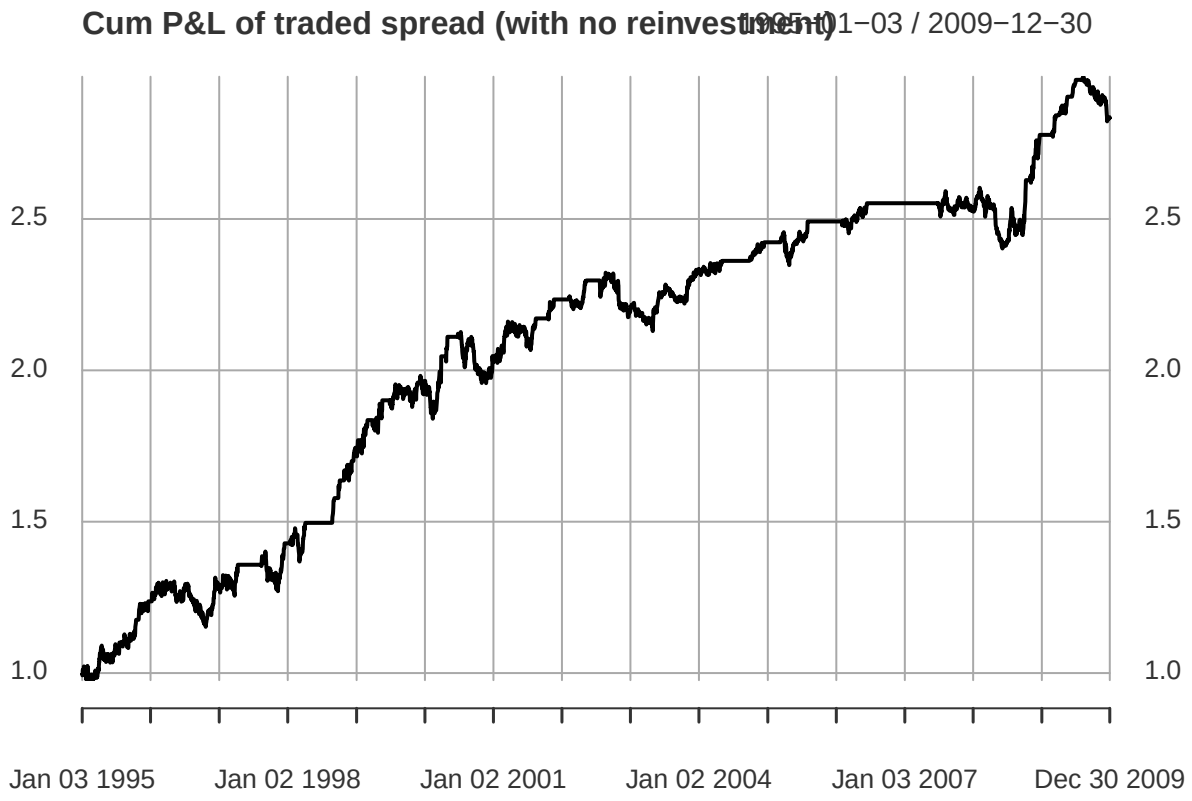
1. similar to portfolio return

```
return <- diff(spread) * lag(signal)
return[is.na(return)] <- 0
colnames(return) <- "return"
plot(return, legend.loc = "topleft")
```



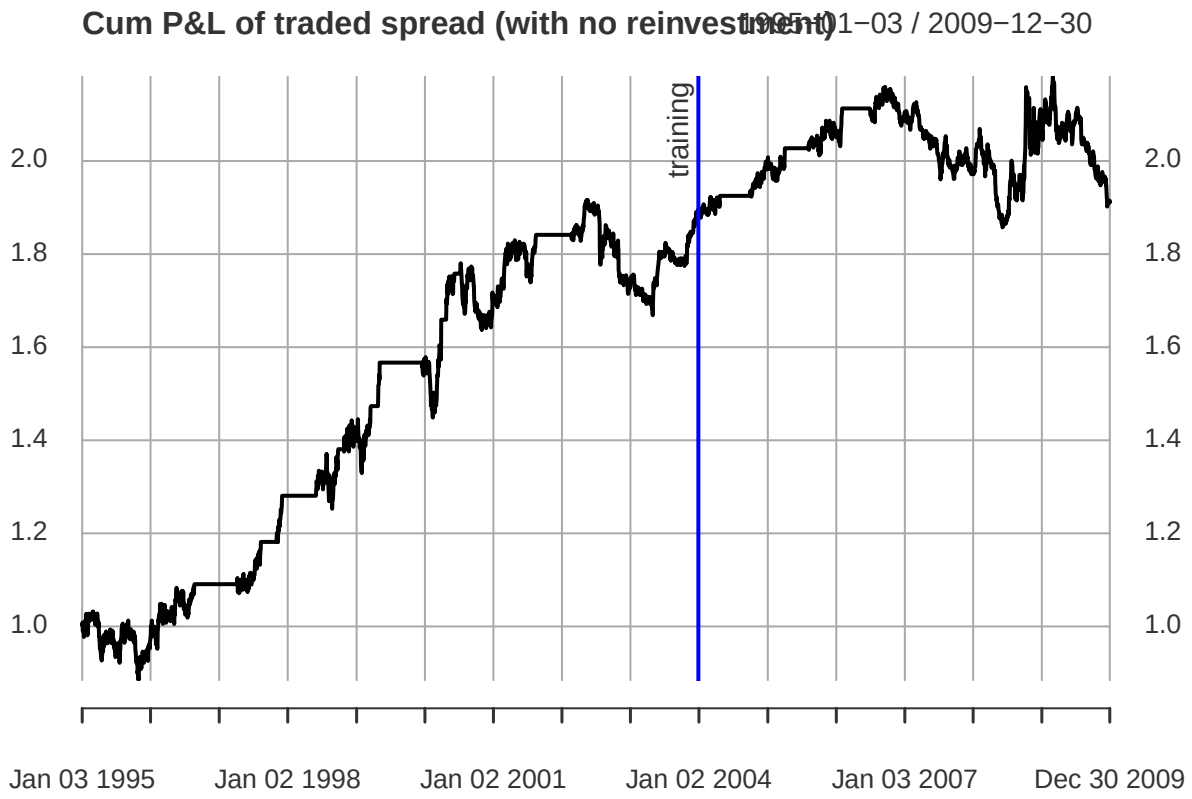
Compute cumulative investment return with no reinvestment

```
plot(1 + cumsum(return), main = "Cum P&L of traded spread (with no reinvestment)")
```



Define a function for futher convenience

```
analyze_spread <- function(logprices, prop_trn = 0.7, threshold = 0.7, legend_loc = "topleft"){
  T_trn <- round(prop_trn*T)
  T_tst <- T - T_trn
  y1 <- logprices[,1]
  y2 <- logprices[,2]
  ls_coeffs <- coef(lm(y1[1:T_trn] ~ y2[1:T_trn]))
  gamma <- ls_coeffs[2]
  spread <- y1 - gamma*y2
  colnames(spread) <- paste(colnames(y1), "-", colnames(y2))
  spread_mean <- mean(spread)
  spread_volatility <- sd(spread)
  z_score <- (spread - spread_mean)/spread_volatility
  signal <- generate_signal(z_score, threshold, -threshold)
  return <- diff(spread) * lag(signal)
  return[is.na(return)] <- 0
  {plot(1 + cumsum(return), main = "Cum P&L of traded spread (with no reinvestment)")
    addEventLines(xts("training", index(y1[T_trn])), lwd = 2, col = "blue", srt = 90, pos = 2 )}
}
analyze_spread(logprices, prop_trn = 0.6)
```



The drawback of this method: Assuming gamma is invariant, and gamma is somewhat related to weights. However if prices changes it definetely leads to the change of weight, which contradictes our invariant gamma assumption

[Unfinished part]

Can we think in this way: we buy one share of y_1, and gamma share of y_2 at the every start of trading that means at every beginning of open position, we rebalance our weights based on our culmulative wealth.

```
return_y1 <- diff(y1)*lag(signal)
return_y2 <- diff(y2)*lag(signal)
```

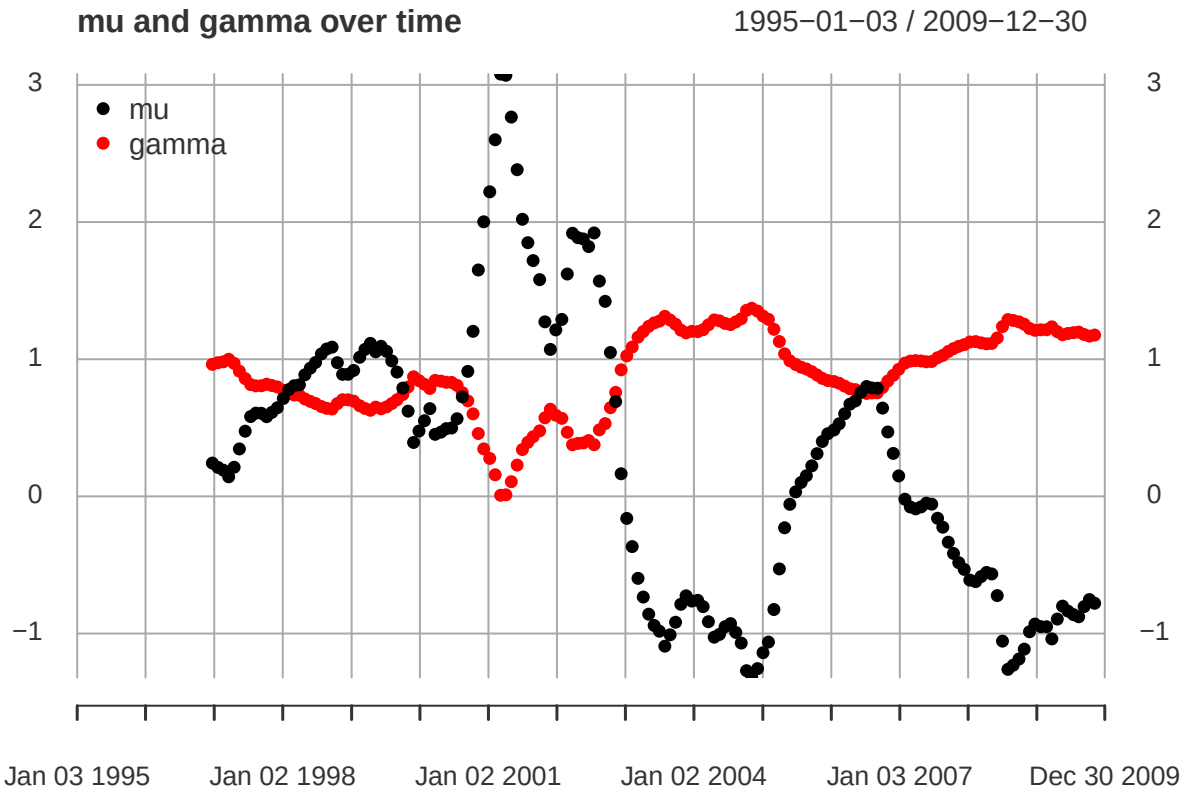
Time variant estimate

Rolling LS

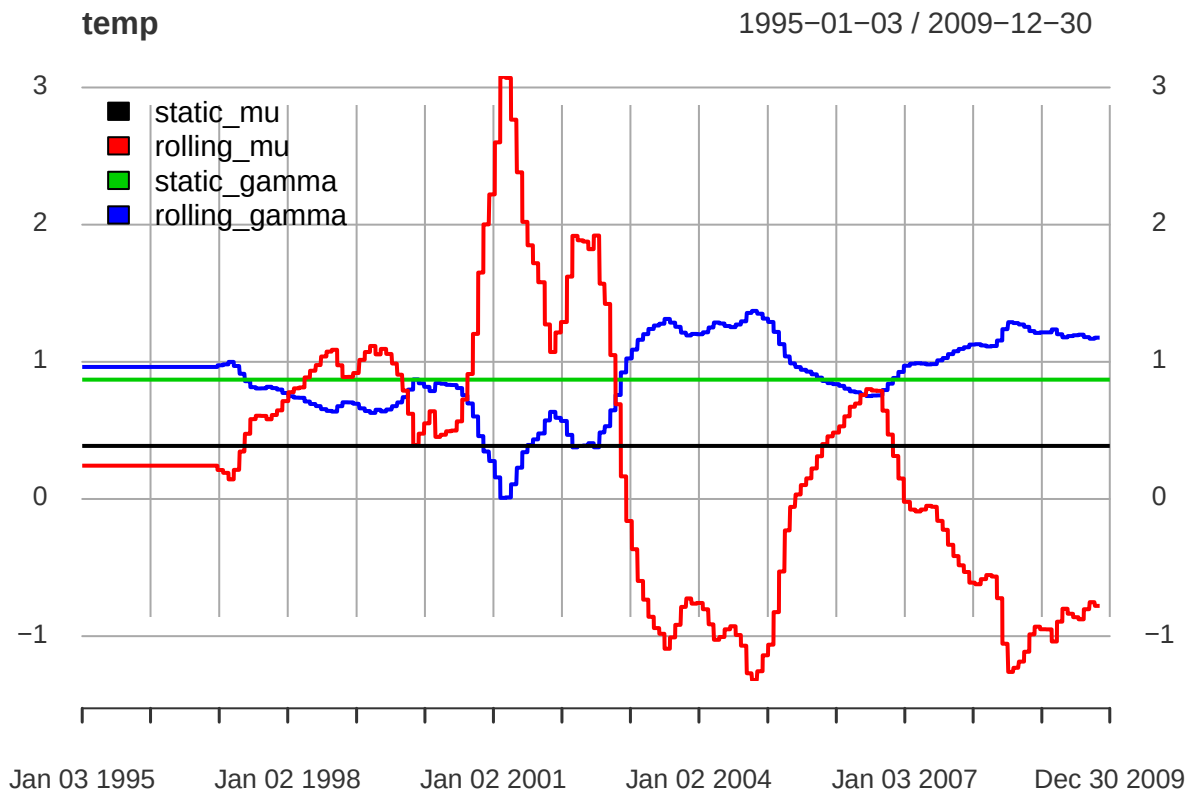
```
T_window <- 500
T_frenq <- 20
t0_update <- seq(from = T_window, to = T - T_frenq, by = T_frenq)
mu_rolling <- gamma_rolling <- xts(rep(NA, T), index(logprices))

for (t0 in t0_update){
  coeffs <- coef(lm(logprices[(t0 - T_window + 1): t0, 1] ~ logprices[(t0 - T_window + 1): t0, 2]))
  mu_rolling[t0 + 1] <- coeffs[1]
  gamma_rolling[t0 + 1] <- coeffs[2]
}
```

```
{plot(cbind(mu_rolling, gamma_rolling), type = "o", pch = 16, main = "mu and gamma over time")
addLegend(legend.loc = "topleft", legend.names = c("mu","gamma"), col = c("black", "red"), pch = c(16,
```




```
mu_rolling <- na.locf(mu_rolling, fromLast = TRUE) #replace NA with the next value
gamma_rolling <- na.locf(gamma_rolling, fromLast = TRUE)
temp <- cbind(mu, mu_rolling, gamma, gamma_rolling)
colnames(temp) <- c("static_mu", "rolling_mu", "static_gamma", "rolling_gamma")
plot(temp, legend.loc = "topleft")
```

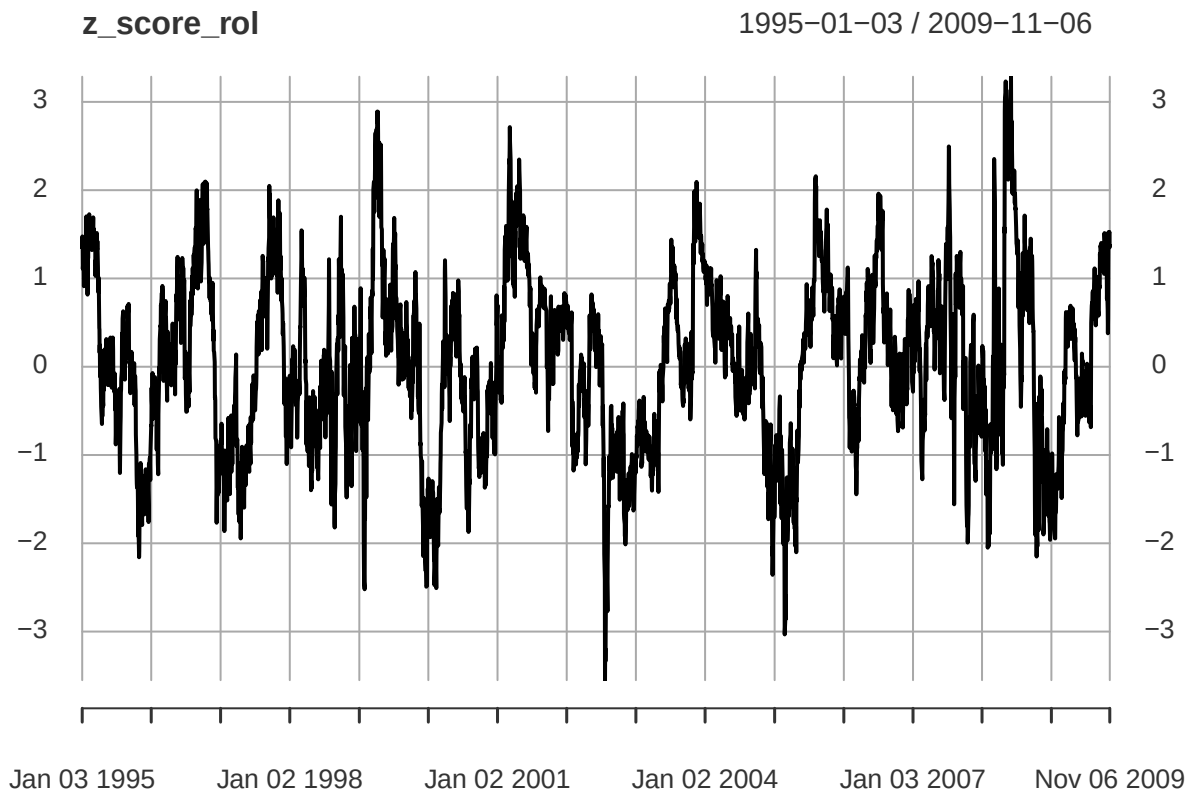


Compute spread based on rolling estimate

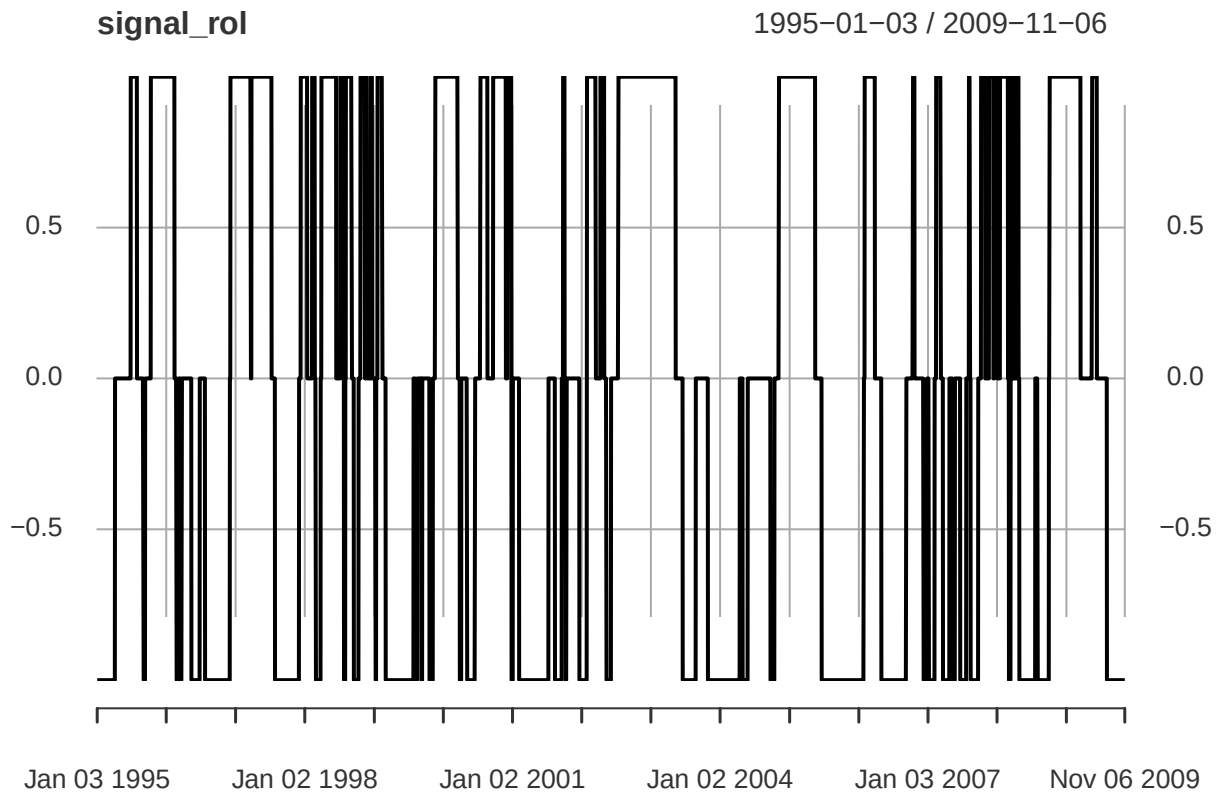
```
spread_rolling <- y1 -gamma_rolling * y2 - mu_rolling
colnames(spread_rolling) <- "spread_rolling"
```

now lets define a function that computes z_score based on rolling window regression

```
library(TTR)
spread_rol <- spread_rolling[!is.na(spread_rolling)]
generate_z_score_rolling <- function(spread_rol, n = 252){
  spread_rol.mean <- EMA(spread_rol, n) # exponentially-weighted mean, giving more weight to recent obs
  spread_rol.mean <- na.locf(spread_rol.mean, fromLast = TRUE)
  spread.demeaned <- spread_rol - spread_rol.mean
  spread_rol.var <- EMA(spread.demeaned^2, n)
  spread_rol.var <- na.locf(spread_rol.var, fromLast = TRUE)
  z_score_rol <- spread.demeaned/sqrt(spread_rol.var)
  return(z_score_rol)
}
z_score_rol <- generate_z_score_rolling(spread_rol, n = 250)
plot(z_score_rol)
```



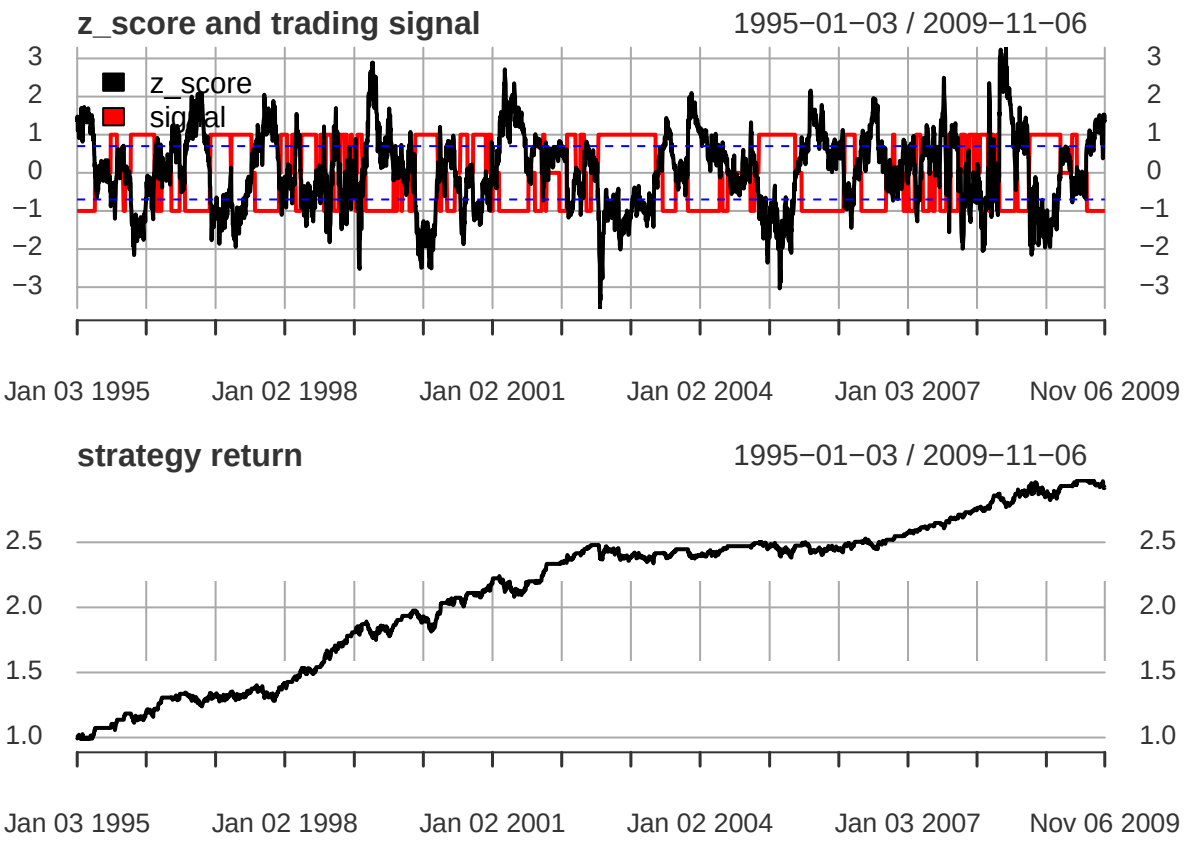

```
signal_rol <- generate_signal(z_score_rol, upper_bound = 0.7, lower_bound = -0.7)
plot(signal_rol) # trade more frequently than the static situation
```




```
port_ret_rol <- diff(spread_rol) * lag(signal_rol) # we assume fix gamma and mu during a trade, however
port_ret_rol[1] <- 0 # change first NA to 0
```

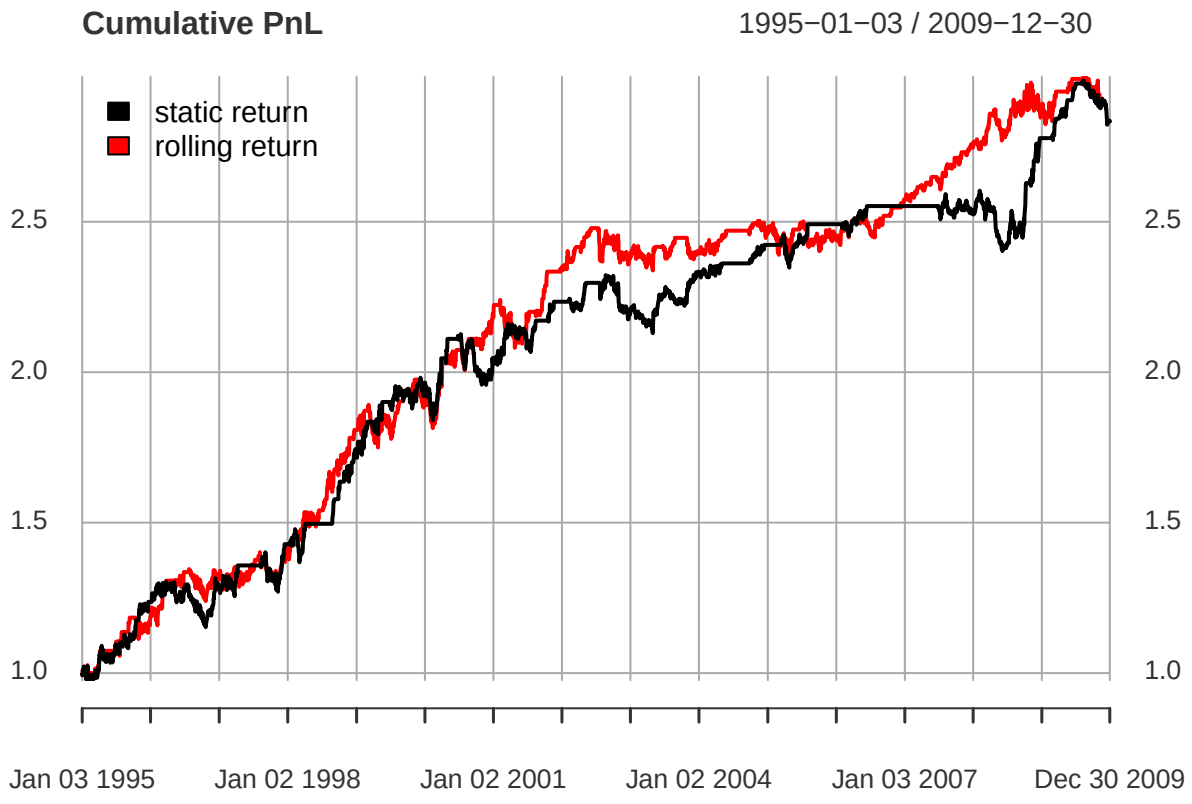
Visulize strategy performance

```
tmp <- cbind(z_score_rol, signal_rol)
colnames(tmp) <- c("z_score", "signal")
par(mfrow = c(2, 1))
{plot(tmp, legend.loc = "topleft", main = "z_score and trading signal")
  lines(xts(rep(0.7, nrow(z_score_rol)), index(z_score_rol)), lty = 2, col = "blue")
  lines(xts(rep(-0.7, nrow(z_score_rol)), index(z_score_rol)), lty = 2, col = "blue")}
plot(1 + cumsum(port_ret_rol), main = "strategy return")
```



Compare static LS with rolling LS

```
par(mfrow = c(1, 1))
tmp <- cbind(1 + cumsum(return), 1 + cumsum(port_ret_rol))
colnames(tmp) <- c("static return", "rolling return")
plot(tmp, legend.loc = "topleft", main = "Cumulative PnL")
```


Cumulative PnL
1995−01−03 / 2009−12−30
static return
rolling return
2.5
2.0
1.5
1.0
Jan 03 1995
Jan 02 1998
Jan 02 2001
Jan 02 2004
Jan 03 2007
Dec 30 2009