Tony Au (Github: restinghouse0203)

Pair Trading with Time-Series Model: BHP vs VALE

Introduction

This project aims to study the multivariate time series error correction statistical arbitrage model on spread pair trading. We specifically analyze the relationship between BHP and VALE, estimating a Vector Error Correction Model (VECM) to capture the cointegration dynamics. The study concludes with an out-of-sample backtest from 2007 to 2023 to evaluate the strategy's robustness in different market conditions.

Project Outline:

1. **Data Preparation:** Loading and transforming daily data (2002-2006).
2. **Preliminary Analysis:** OLS estimation and residual analysis to check for stationarity.
3. **Cointegration Analysis:** Johansen Test and VECM estimation.
4. **Statistical Arbitrage Strategy:**
   - Methodology & Trading Rules.
   - In-Sample Implementation & Performance (vs Benchmark).
   - Out-of-Sample Backtest (2007-2023) with robust performance metrics.

1. Data Preparation

```r
# Load necessary libraries and data
library(urca)
library(zoo) # Added for time series handling
bhp = read.table("d-bhp0206.txt", header=T)
vale = read.table("d-vale0206.txt", header=T)

# Create Date object for plotting
dates <- as.Date(paste(bhp$year, bhp$Mon, bhp$day, sep="-"), format="%Y-%m-%d")
```

We perform a logarithmic transformation on the adjusted closing prices to stabilize variance.

```r
# Log transform the adj. close price
bhp = log(bhp[,9])
vale = log(vale[,9])
```

2. Preliminary Analysis

2.1 Least-Squares Estimation

We begin by estimating the static relationship between BHP and VALE using Ordinary Least Squares (OLS).

```
m1 = lm(bhp~vale)
summary(m1)
```

```
##
## Call:
## lm(formula = bhp ~ vale)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.151818 -0.028265  0.003121  0.029803  0.147105
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.822648   0.003662   497.7   <2e-16 ***
## vale        0.716664   0.002354   304.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04421 on 944 degrees of freedom
## Multiple R-squared:  0.9899, Adjusted R-squared:  0.9899
## F-statistic: 9.266e+04 on 1 and 944 DF,  p-value: < 2.2e-16
```

The estimated static relationship is:

$$\widehat{\text{bhp}} = 1.823 + 0.717 \cdot \text{vale} + \varepsilon, \quad \varepsilon \sim N(0, 0.044^2)$$

Interpretation: The coefficient indicates that for every 1% increase in VALE, BHP increases by approximately 0.717% on average. The model explains 98.99% of the variability in BHP ($R^2 = 0.9899$), suggesting a strong comovement.

2.2 Residual Analysis (AR(2))

We analyze the residuals of the OLS model to check for stationarity and serial correlation.

```
library(tseries)
wt = m1$residuals
m3 = arima(wt,order=c(2,0,0),include.mean = F)
m3
```

```
##
## Call:
## arima(x = wt, order = c(2, 0, 0), include.mean = F)
##
## Coefficients:
##          ar1     ar2
##       0.8051  0.1219
## s.e.  0.0322  0.0325
```

```
##
## sigma^2 estimated as 0.0003326:  log likelihood = 2444.76,  aic = -4883.51
```

```
# ADF test for wt stationarity
adf.test(wt, k=2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  wt
## Dickey-Fuller = -6.0306, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Characteristic roots decomposition
p1=c(1,-m3$coef)
print("The characteristic roots are")
```

```
## [1] "The characteristic roots are"
```

```
1/Mod(polyroot(p1))
```

```
## [1] 0.9353661 0.1302870
```

The error process is modeled as:

$$w_t = 0.8051\, w_{t-1} + 0.1219\, w_{t-2} + \varepsilon_t = (1 - 0.935B)(1 - 0.13B)w_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, 0.0003326)$$

Interpretation: The modulus of both characteristic roots is less than 1, and the Augmented Dickey-Fuller (ADF) test rejects the null hypothesis of a unit root. This confirms that the residual series w_t is stationary, a prerequisite for cointegration. The current value of w_t depends strongly on its immediate lag (0.805) and weakly on the second lag (0.122).

3. Cointegration Analysis

3.1 Johansen Cointegration Test

We apply the Johansen procedure to test for the number of cointegrating relationships.

```
# install.packages("HDTSA")
library(HDTSA)
xt = cbind(bhp,vale)

# Cointegrating relations
cot=ca.jo(xt,ecdet = "const",type = "trace",K=2,spec = "transitory")
summary(cot)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1]  4.148282e-02  8.206470e-03 -4.210618e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  7.78  7.52  9.24 12.97
## r = 0  | 47.77 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             bhp.l1     vale.l1   constant
## bhp.l1    1.000000  1.0000000  1.000000
## vale.l1  -0.717704 -0.7327542  2.047274
## constant -1.828460 -1.5411890 -5.712629
##
## Weights W:
## (This is the loading matrix)
##
##             bhp.l1      vale.l1       constant
## bhp.d  -0.06731196 0.004568985 -7.703496e-18
## vale.d  0.02545606 0.007541565  3.627982e-18
```

```
co1=ca.jo(xt,ecdet = "const",type = "eigen",K=2,spec = "transitory")
summary(co1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1]  4.148282e-02  8.206470e-03 -4.210618e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  7.78  7.52  9.24 12.97
## r = 0  | 40.00 13.75 15.67 20.20
```

```
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              bhp.l1     vale.l1   constant
## bhp.l1     1.000000  1.0000000  1.000000
## vale.l1  -0.717704 -0.7327542  2.047274
## constant -1.828460 -1.5411890 -5.712629
##
## Weights W:
## (This is the loading matrix)
##
##              bhp.l1      vale.l1       constant
## bhp.d  -0.06731196 0.004568985 -7.703496e-18
## vale.d  0.02545606 0.007541565  3.627982e-18
```

Interpretation: Both the trace and maximal eigenvalue tests indicate a cointegration rank of $r = 1$. This confirms a single long-run equilibrium relationship:

$$\text{bhp} \approx 0.718 \times \text{vale} + 1.83$$

This relationship is consistent with the initial OLS estimate.

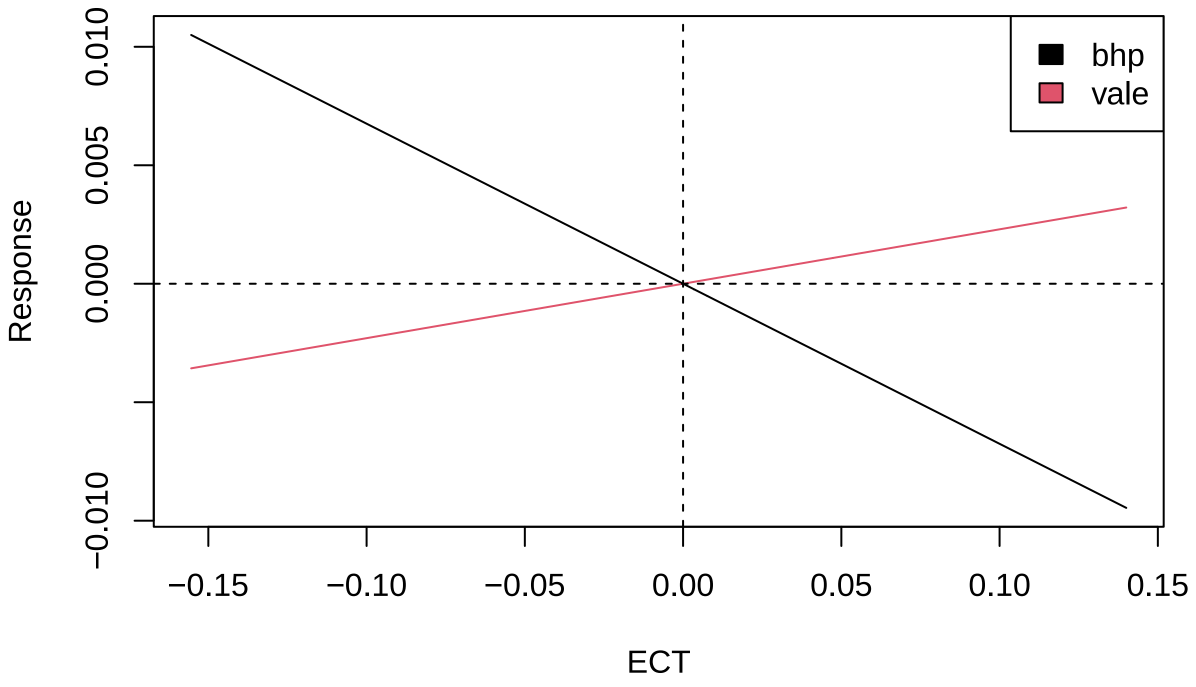
3.2 Vector Error Correction Model (VECM)

We estimate a VECM to understand the adjustment dynamics.

```
# install.packages("tsDyn")
library(tsDyn)
library(vars)

# Data setup
xt <- ts(cbind(bhp = bhp, vale = vale))

# Estimate VECM directly
vecm <- VECM(xt,
            lag       = 2,              # lags in differences
            r         = 1,              # cointegrating rank
            include   = "const",        # constant inside cointegration relation
            estim     = "ML",           # estimation method
            LRinclude = "const")        # constant location

# Summary
summary(vecm)
```

```
## #############
## ###Model VECM
## #############
## Full sample size: 946    End sample size: 943
## Number of variables: 2   Number of estimated slope parameters 10
## AIC -14851.66    BIC -14798.32   SSR 0.8210436
## Cointegrating vector (estimated by ML):
##    bhp       vale     const
## r1   1 -0.7172022 -1.829477
##
##
##                 ECT                 bhp -1              vale -1
## Equation bhp  -0.0675(0.0148)*** -0.1102(0.0374)**  0.0724(0.0325)*
## Equation vale 0.0230(0.0171)     0.0662(0.0433)     0.0493(0.0376)
##                 bhp -2              vale -2
## Equation bhp  -0.0062(0.0371)     -0.0153(0.0322)
## Equation vale 0.0134(0.0430)      -0.0688(0.0373).
```

```
# Plot of the cointegration relation
plot_ECT(vecm,
         add.legend      = TRUE,
         legend.location = "topright")
```



Model Structure

Matrix form:

$$\begin{pmatrix}\Delta \text{bhp}_t \\ \Delta \text{vale}_t\end{pmatrix} = \alpha(\beta' z_{t-1}) + \Gamma_1 \begin{pmatrix}\Delta \text{bhp}_{t-1} \\ \Delta \text{vale}_{t-1}\end{pmatrix} + \Gamma_2 \begin{pmatrix}\Delta \text{bhp}_{t-2} \\ \Delta \text{vale}_{t-2}\end{pmatrix} + \varepsilon_t$$

Matrix form (with parameters):

Let $w_t = \text{BHP}_t - 0.717 \cdot \text{VALE}_t$ and $\mu = E(w_t) = 1.829$.

$$\begin{pmatrix} \Delta\text{bhp}_t \\ \Delta\text{vale}_t \end{pmatrix} = \begin{pmatrix} -0.0675 \\ 0.0230 \end{pmatrix} (w_{t-1} - 1.829) + \begin{pmatrix} -0.1102 & 0.0724 \\ 0.0662 & 0.0493 \end{pmatrix} \begin{pmatrix} \Delta\text{bhp}_{t-1} \\ \Delta\text{vale}_{t-1} \end{pmatrix} + \begin{pmatrix} -0.0062 & -0.0153 \\ 0.0134 & -0.0688 \end{pmatrix} \begin{pmatrix} \Delta\text{bhp}_{t-2} \\ \Delta\text{vale}_{t-2} \end{pmatrix} + \varepsilon_t$$

Interpretation: The error correction term for BHP (-0.0675) is statistically significant, indicating that BHP adjusts towards the equilibrium at a speed of approximately 6.75% per period. In contrast, the adjustment coefficient for VALE is small and insignificant, suggesting that VALE is weakly exogenous and acts as the driver in this pair.

4. Statistical Arbitrage Strategy

4.1 Methodology

We implement a mean-reversion strategy based on the spread $w_t = \text{BHP}_t - 0.717 \cdot \text{VALE}_t$, with mean $\mu = 1.829$ and standard deviation σ_w.

Trading Rules:

- **Upper Threshold:** $\mu + \sigma_w$
- **Lower Threshold:** $\mu - \sigma_w$
- **Signal:**
  - If $w_t > \mu + \sigma_w$: Short 1 unit of BHP, Long 0.717 units of VALE (betting spread decreases).
  - If $w_t < \mu - \sigma_w$: Long 1 unit of BHP, Short 0.717 units of VALE (betting spread increases).
- **Exit:** When w_t reverts to the mean μ.

4.2 Implementation (In-Sample)

We compute the spread, generate trading signals, and compare performance against a Buy-and-Hold benchmark.

```r
# 1. Compute spread w_t
gamma <- 0.717 # w_t := bhp_t - \gamma*vale_t
mu <- 1.829 # E(w_t)
w_t <- xt[, "bhp"] - gamma * xt[, "vale"]

# Standard deviation of w_t
sd_w <- sd(w_t)

# Residuals from VECM
res <- residuals(vecm)
sd_eps_bhp <- sd(res[, 1])
sd_eps_vale <- sd(res[, 2])

# Output statistics
cat("SD of w_t:", sd_w, "\n")
```

```
## SD of w_t: 0.04418214
```

```
cat("SD of  _bhp:", sd_eps_bhp, "\n")
```

```
## SD of  _bhp: 0.01926604
```

```
cat("SD of  _vale:", sd_eps_vale, "\n")
```

```
## SD of  _vale: 0.02224961
```

```
# 2. Trading Thresholds
upper <- mu + sd_w
lower <- mu - sd_w

# 3. Generate Signals (In-Sample)
positions <- rep(0, length(w_t))
for (t in 2:length(w_t)) {
  if (w_t[t-1] > upper) positions[t] <- -1
  else if (w_t[t-1] < lower) positions[t] <- 1
  else if (positions[t-1] != 0 && abs(w_t[t-1] - mu) < 0.1 * sd_w) positions[t] <- 0
  else positions[t] <- positions[t-1]
}

# 4. Calculate Returns
# In-sample log returns
ret_bhp <- diff(xt[, "bhp"]) # xt is already log prices
ret_vale <- diff(xt[, "vale"])

# Strategy Returns
strat_ret <- positions[2:length(positions)] * (ret_bhp - gamma * ret_vale)
strat_ret <- na.omit(strat_ret)

# Benchmark Returns (50/50 Buy and Hold)
bench_ret <- 0.5 * ret_bhp + 0.5 * ret_vale
bench_ret <- na.omit(bench_ret)

# 5. Performance Metrics
# Strategy
cum_pnl <- cumsum(strat_ret)
pnl <- sum(strat_ret)
max_dd <- min(cum_pnl - cummax(cum_pnl))
sharpe <- mean(strat_ret) / sd(strat_ret) * sqrt(252)
hit_rate <- sum(strat_ret > 0) / length(strat_ret[strat_ret != 0])

# Benchmark
cum_bench <- cumsum(bench_ret)
```

```r
pnl_bench <- sum(bench_ret)
sharpe_bench <- mean(bench_ret) / sd(bench_ret) * sqrt(252)

cat("\nIn-Sample Performance (2002-2006):\n")
```

```
##
## In-Sample Performance (2002-2006):
```

```r
cat("Strategy PnL:", round(pnl, 4), " | Benchmark PnL:", round(pnl_bench, 4), "\n")
```

```
## Strategy PnL: 1.9882  | Benchmark PnL: 1.6749
```

```r
cat("Strategy MaxDD:", round(max_dd, 4), "\n")
```

```
## Strategy MaxDD: -0.1297
```

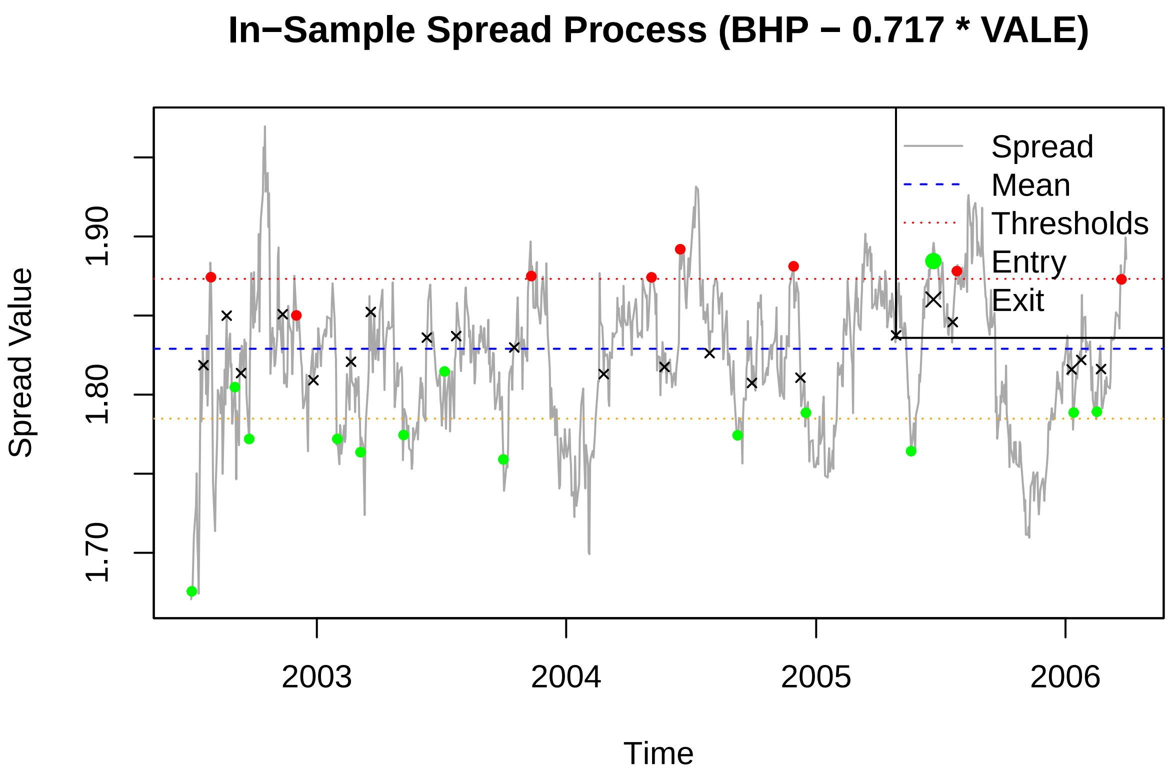
```r
cat("Strategy Sharpe:", round(sharpe, 4), " | Benchmark Sharpe:", round(sharpe_bench, 4), "\n")
```

```
## Strategy Sharpe: 2.1882  | Benchmark Sharpe: 1.5652
```

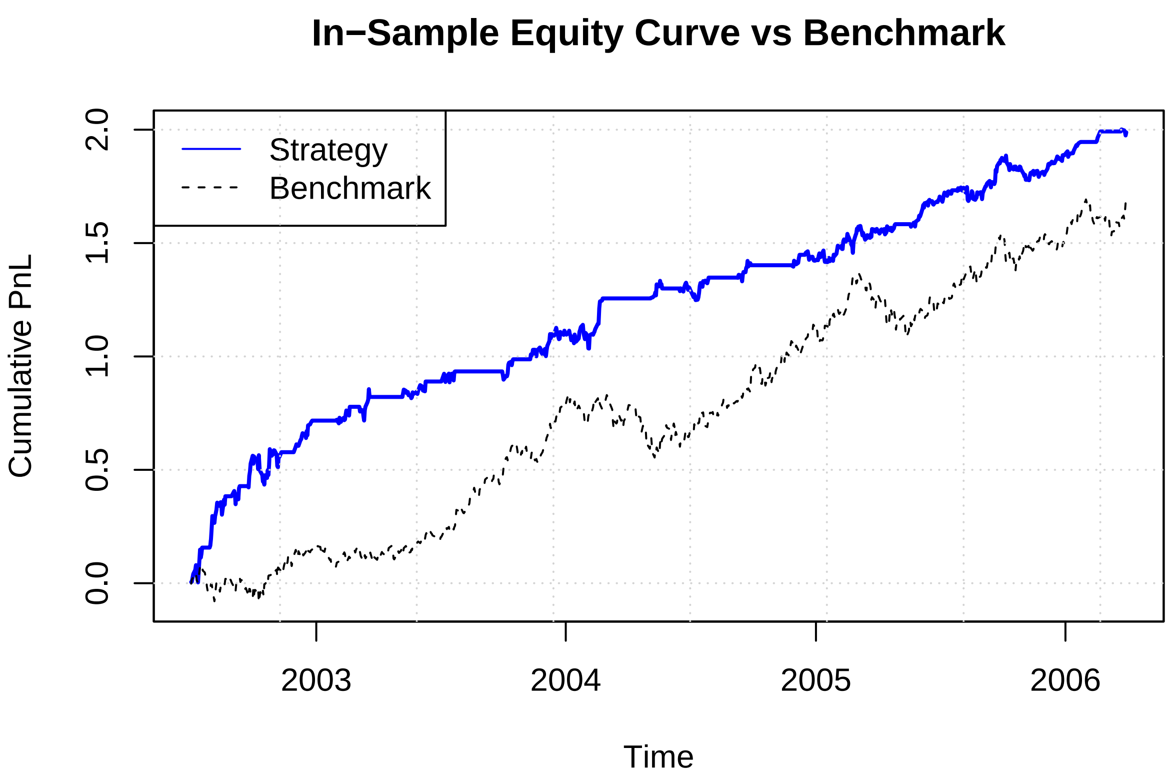
```r
cat("Hit Rate:", round(hit_rate, 4), "\n")
```

```
## Hit Rate: 0.5742
```

```r
# 6. Visualization
# Create zoo objects for plotting with dates
# Note: w_t length matches dates, but returns are length-1
w_zoo <- zoo(w_t, order.by = dates)
cum_pnl_zoo <- zoo(cum_pnl, order.by = dates[-1])
cum_bench_zoo <- zoo(cum_bench, order.by = dates[-1])

# A. Spread Process
plot(w_zoo, type="l", main="In-Sample Spread Process (BHP - 0.717 * VALE)", ylab="Spread Value", xlab="
abline(h=mu, col="blue", lty=2)
abline(h=upper, col="red", lty=3)
abline(h=lower, col="orange", lty=3)

# Add Signals
time_index <- index(w_zoo)
long_entries <- which(positions == 1 & c(0, positions[-length(positions)]) == 0)
short_entries <- which(positions == -1 & c(0, positions[-length(positions)]) == 0)
exits <- which(positions == 0 & c(0, positions[-length(positions)]) != 0)

if(length(long_entries) > 0) points(time_index[long_entries], w_t[long_entries], col="green", pch=19, cex=0.6)
if(length(short_entries) > 0) points(time_index[short_entries], w_t[short_entries], col="red", pch=19, cex=0.6
if(length(exits) > 0) points(time_index[exits], w_t[exits], col="black", pch=4, cex=0.6)

legend("topright", c("Spread", "Mean", "Thresholds", "Entry", "Exit"),
      col=c("darkgray","blue","red", "green", "black"), lty=c(1,2,3,NA,NA), pch=c(NA,NA,NA,19,4))
```



```
# B. Equity Curve
plot(cum_pnl_zoo, type = "l", col = "blue", lwd = 2,
     main = "In-Sample Equity Curve vs Benchmark",
     xlab = "Time", ylab = "Cumulative PnL", ylim=range(c(cum_pnl, cum_bench)))
lines(cum_bench_zoo, col="black", lty=2)
legend("topleft", c("Strategy", "Benchmark"), col=c("blue", "black"), lty=c(1,2))
grid()
```



4.3 Out-of-Sample Backtest Results (2007-2023)

We perform out-of-sample backtesting using data from 2007 to 2023 obtained from Yahoo Finance.

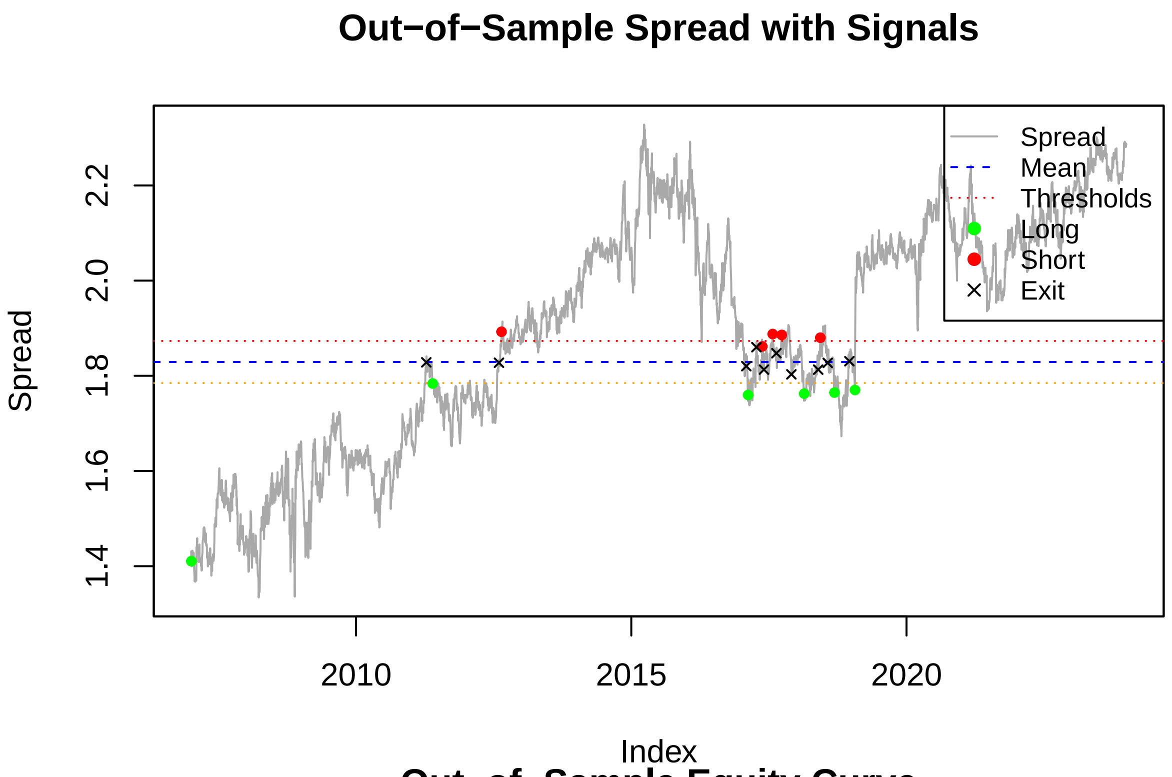
```
# --- PART 1: FETCH DATA ---
#install.packages("quantmod")
library(quantmod)

s_date <- "2007-01-01"
e_date <- "2023-12-31"
bhp_file <- "BHP_OS.csv"
vale_file <- "VALE_OS.csv"

BHP <- as.xts(read.zoo(bhp_file, header = TRUE, sep = ",", format = "%Y-%m-%d"))
VALE <- as.xts(read.zoo(vale_file, header = TRUE, sep = ",", format = "%Y-%m-%d"))
```

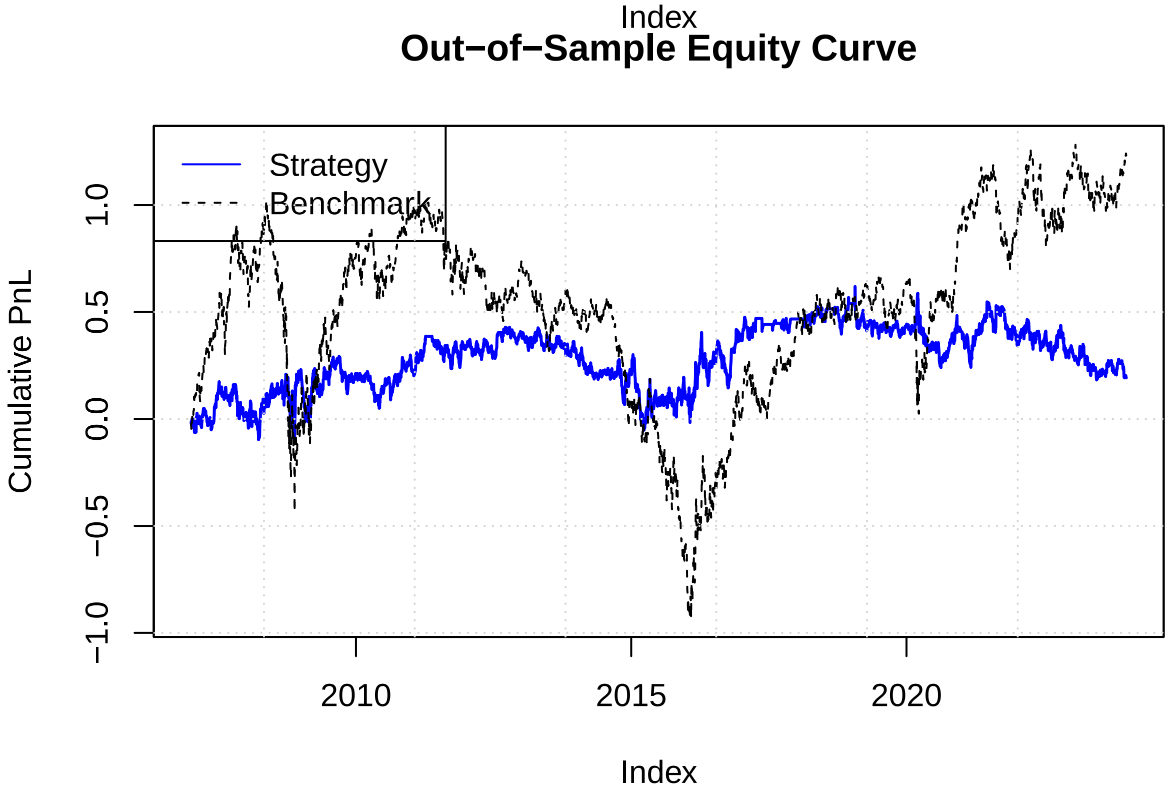
```
# --- PART 2: STRATEGY FUNCTIONS ---

# Function to preprocess and calculate spread
calculate_spread_os <- function(BHP_data, VALE_data, gamma) {
  if (!exists("BHP_data") || !exists("VALE_data")) return(NULL)

  bhp_adj <- Ad(BHP_data)
  vale_adj <- Ad(VALE_data)
```

```r
  pair <- merge(bhp_adj, vale_adj, all = FALSE)
  colnames(pair) <- c("bhp", "vale")

  # Log transformation
  log_pair <- log(pair)

  # Calculate Spread
  w_os <- log_pair$bhp - gamma * log_pair$vale

  list(w_os = w_os, log_pair = log_pair)
}

# Function to generate trading signals
generate_signals_os <- function(w_os, mu, sd_w) {
  upper <- mu + sd_w
  lower <- mu - sd_w

  positions <- rep(0, length(w_os))
  spread_val <- as.numeric(w_os)

  for (t in 2:length(w_os)) {
    if (spread_val[t-1] > upper) {
      positions[t] <- -1
    } else if (spread_val[t-1] < lower) {
      positions[t] <- 1
    } else if (positions[t-1] != 0 && abs(spread_val[t-1] - mu) < 0.1 * sd_w) {
      positions[t] <- 0
    } else {
      positions[t] <- positions[t-1]
    }
  }
  return(positions)
}

# Function to calculate performance metrics
calculate_metrics_os <- function(positions, log_pair, gamma) {
  ret_bhp <- diff(log_pair$bhp)
  ret_vale <- diff(log_pair$vale)

  # Strategy Returns
  strat_ret <- positions[2:length(positions)] * (ret_bhp - gamma * ret_vale)
  strat_ret <- na.omit(strat_ret)

  # Benchmark Returns (50/50 Equal Weight)
  bench_ret <- 0.5 * ret_bhp + 0.5 * ret_vale
  bench_ret <- na.omit(bench_ret)

  # Metrics Strategy
  cum_pnl <- cumsum(strat_ret)
  pnl <- sum(strat_ret)
  max_dd <- min(cum_pnl - cummax(cum_pnl))
  sharpe <- mean(strat_ret) / sd(strat_ret) * sqrt(252)
  hit_rate <- sum(strat_ret > 0) / length(strat_ret[strat_ret != 0])

  # Metrics Benchmark
  cum_bench <- cumsum(bench_ret)
  pnl_bench <- sum(bench_ret)
  sharpe_bench <- mean(bench_ret) / sd(bench_ret) * sqrt(252)

  list(
    strat_ret = strat_ret, bench_ret = bench_ret,
    cum_pnl = cum_pnl, cum_bench = cum_bench,
    pnl = pnl, pnl_bench = pnl_bench,
    max_dd = max_dd,
    sharpe = sharpe, sharpe_bench = sharpe_bench,
    hit_rate = hit_rate
  )
}
```

```
# --- PART 3: EXECUTION & VISUALIZATION ---

if (exists("BHP") && exists("VALE")) {
  # Parameters (from In-Sample)
  gamma_is <- 0.717
  mu_is <- 1.829
  # sd_w from section 4.2 needs to be passed or hardcoded if not available globally in knitting context.
  # Assuming sd_w is available from previous chunks. If not, use approximation from text:
  # sd_w value from 4.2 output. Let's assume it's available as 'sd_w'.

  # 1. Process Data
  data_os <- calculate_spread_os(BHP, VALE, gamma_is)
  w_os <- data_os$w_os

  # 2. Generate Signals
  positions_os <- generate_signals_os(w_os, mu_is, sd_w)

  # 3. Calculate Metrics
  res_os <- calculate_metrics_os(positions_os, data_os$log_pair, gamma_is)
  # 4. Print Results
  cat(”Out-of-Sample Results (2007-2023):\n”)
  cat(”Strategy PnL:”, round(res_os$pnl, 4), ” | Benchmark PnL:”, round(res_os$pnl_bench, 4), ”\n”)
  cat(”Strategy MaxDD:”, round(res_os$max_dd, 4), ”\n”)
  cat(”Strategy Sharpe:”, round(res_os$sharpe, 4), ” | Benchmark Sharpe:”, round(res_os$sharpe_bench, 4), ”\
  cat(”Hit Rate:”, round(res_os$hit_rate, 4), ”\n”)

  # 5. Visualization
  # A. Spread
  upper <- mu_is + sd_w
  lower <- mu_is - sd_w

  plot(as.zoo(w_os), main=”Out-of-Sample Spread with Signals”, ylab=”Spread”, col=”darkgray”)
  abline(h=mu_is, col=”blue”, lty=2)
  abline(h=upper, col=”red”, lty=3)
  abline(h=lower, col=”orange”, lty=3)

  # Add signal points
  time_index <- index(as.zoo(w_os))
  long_entries <- which(positions_os == 1 & c(0, positions_os[-length(positions_os)]) == 0)
  short_entries <- which(positions_os == -1 & c(0, positions_os[-length(positions_os)]) == 0)
  exits <- which(positions_os == 0 & c(0, positions_os[-length(positions_os)]) != 0)

  if(length(long_entries)>0) points(time_index[long_entries], w_os[long_entries], col=”green”, pch=19, cex=0.6
  if(length(short_entries)>0) points(time_index[short_entries], w_os[short_entries], col=”red”, pch=19, cex=0.
  if(length(exits)>0) points(time_index[exits], w_os[exits], col=”black”, pch=4, cex=0.6)

  legend(”topright”, legend=c(”Spread”, ”Mean”, ”Thresholds”, ”Long”, ”Short”, ”Exit”),
         col=c(”darkgray”, ”blue”, ”red”, ”green”, ”red”, ”black”),
         lty=c(1, 2, 3, NA, NA, NA), pch=c(NA, NA, NA, 19, 19, 4), cex=0.8)

  # B. Equity Curve
  plot(as.zoo(res_os$cum_pnl), main=”Out-of-Sample Equity Curve”, ylab=”Cumulative PnL”, col=”blue”, lwd
       ylim=range(c(res_os$cum_pnl, res_os$cum_bench)))
  lines(as.zoo(res_os$cum_bench), col=”black”, lty=2)
  legend(”topleft”, legend=c(”Strategy”, ”Benchmark”), col=c(”blue”, ”black”), lty=c(1, 2))
  grid()

} else {
  print(”Data not available.”)
}
```

```
## Out-of-Sample Results (2007-2023):
## Strategy PnL: 0.1922  | Benchmark PnL: 1.2329
## Strategy MaxDD: -0.4817
## Strategy Sharpe: 0.0462  | Benchmark Sharpe: 0.1733
## Hit Rate: 0.5009
```





Interpretation of Out-of-Sample Results:

The out-of-sample performance (2007-2023) is significantly worse than the in-sample period, highlighting the dangers of using static model parameters over long horizons.

1. **Poor Performance Metrics:**
   - **PnL (0.1922) & Sharpe (0.0462):** The strategy yielded almost zero risk-adjusted return over 16 years. This indicates the cointegration relationship identified in 2002-2006 effectively broke down or shifted regimes.
   - **Max Drawdown (-0.4817):** A 48% drawdown is catastrophic for a market-neutral strategy. This was likely caused by the spread diverging far beyond the fixed thresholds ($\mu \pm \sigma$) without reverting, forcing the strategy to hold losing positions for extended periods.
   - **Hit Rate (~50%):** A hit rate near 50% implies the entry signals were no better than a coin flip, further confirming that the "mean" $\mu = 1.829$ was no longer the true equilibrium.
2. **Market Shocks & Structural Breaks:**
   - **2008 Financial Crisis:** Correlations often approach 1 during crises, but spreads can widen unpredictably due to liquidity constraints.
   - **Idiosyncratic Shocks (Vale):** Vale suffered massive dam collapses in **2015 (Mariana)** and **2019 (Brumadinho)**. These events caused massive, permanent structural breaks in Vale's price that were unrelated to general iron ore market dynamics (which drive BHP). The model, expecting a reversion to the 2002-2006 mean, would have taken aggressive losing positions against these drifts.
3. **Improvements:**
   - **Rolling Window Calibration:** Instead of fixed parameters, recalibrate γ and μ using a rolling window (e.g., 1-year lookback) to adapt to new market regimes.
   - **Stop-Loss:** Implement a hard stop-loss (e.g., if spread deviates $> 3\sigma$) to close positions when the cointegration assumption is clearly violated.
   - **Kalman Filter:** Use a state-space model to dynamically estimate the time-varying hedge ratio γ_t and mean μ_t.