Solutions to Exercises

Portfolio Optimization: Theory and Application
Chapter 15 – Pairs Trading Portfolios

Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.*
Cambridge University Press.

portfoliooptimizationbook.com

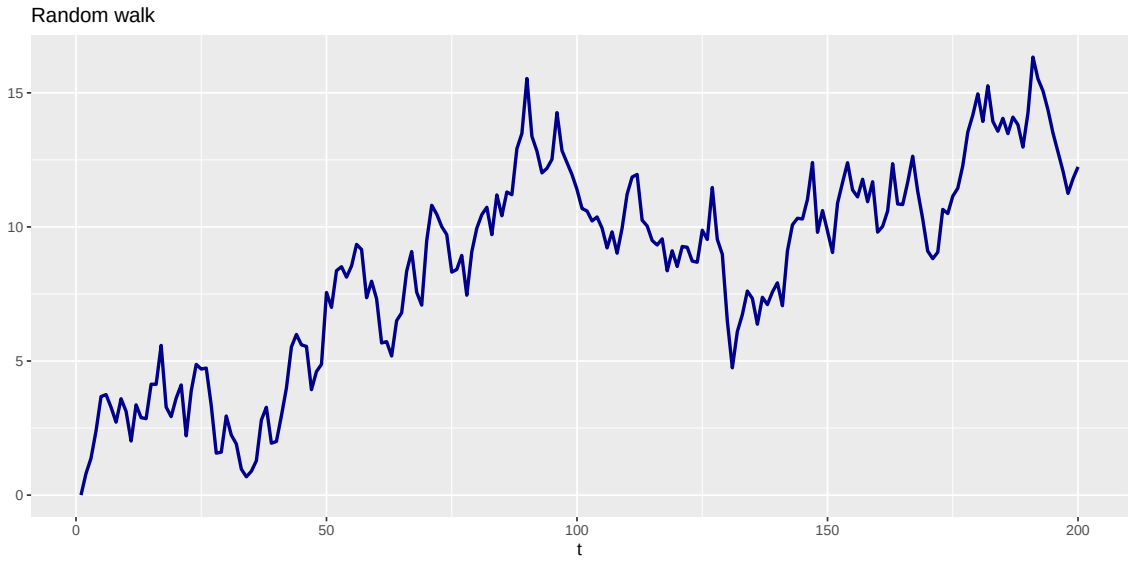
Exercise 15.1: Mean reversion

a. Generate a random walk and plot it. Is it stationary? Does it revert to the mean?
b. Generate an AR(1) sequence with autoregressive coefficient less than 1 and plot it. Is it stationary? Does it revert to the mean?
c. Change the autoregressive coefficient of the AR(1) model and observe how the strength of the mean reversion changes.

Solution

a. Random walk:

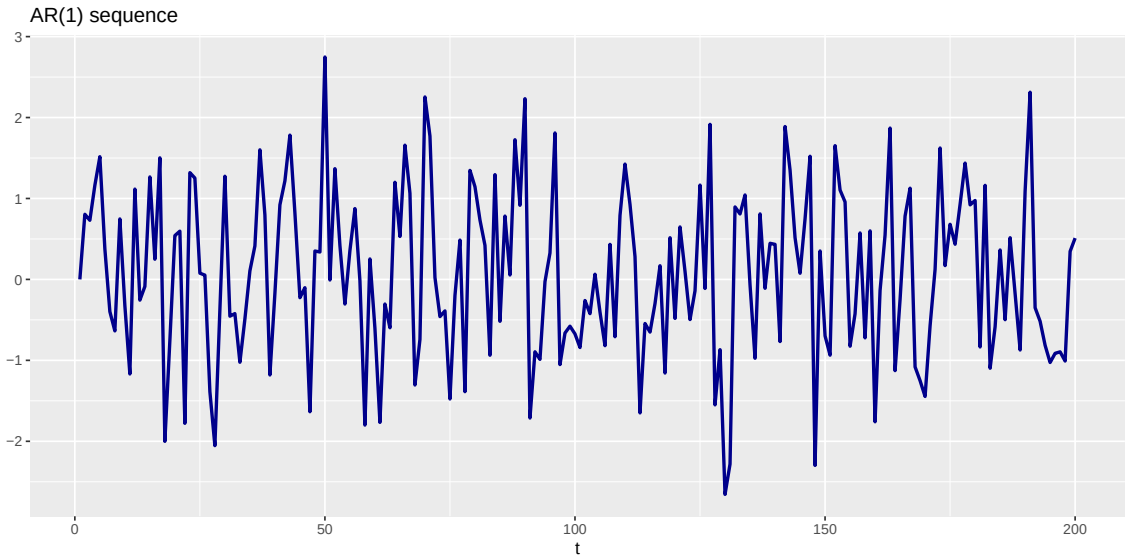
```
library(ggplot2)

T <- 200
set.seed(421)
y <- rep(0, T)
for (i in 2:T)
  y[i] <- y[i-1] + rnorm(1)

data.frame("t" = 1:T, "y" = y) |>
  ggplot(aes(x = t, y = y)) +
  geom_line(linewidth = 1, color = "darkblue") +
  labs(title = "Random walk", y = NULL)
```



It does not revert to the mean. A random walk is a nonstationary time series with unit root.

b. AR(1) sequence with $\rho = 0.2$:

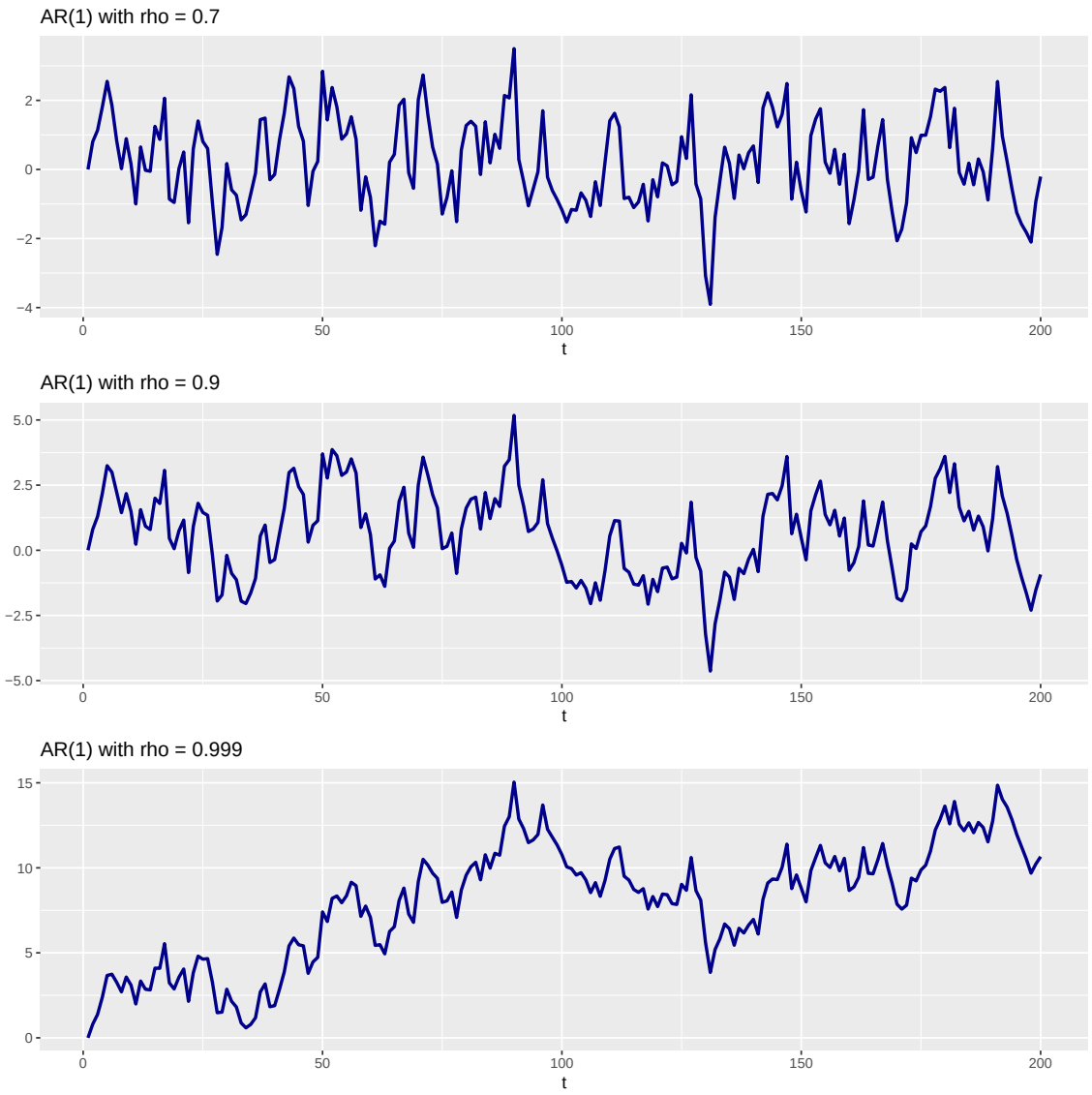
```
T <- 200
set.seed(421)
y <- rep(0, T)
for (i in 2:T)
  y[i] <- 0.2*y[i-1] + rnorm(1)

data.frame("t" = 1:T, "y" = y) |>
  ggplot(aes(x = t, y = y)) +
  geom_line(linewidth = 1, color = "darkblue") +
  labs(title = "AR(1) sequence", y = NULL)
```



It is unit-root stationary because $|\rho| < 1$ and it reverts to the mean.

c. AR(1) sequence with different values of ρ:

```
library(ggplot2)
library(patchwork)

rho_sweep <- c(0.7, 0.9, 0.999)
p_list <- list()

for (rho in rho_sweep) {
  # Generate data
  T <- 200
  set.seed(421)
  y <- rep(0, T)
  for (i in 2:T)
    y[i] <- rho*y[i-1] + rnorm(1)

  # Generate plot
  p <- data.frame("t" = 1:T, "y" = y) |>
    ggplot(aes(x = t, y = y)) +
    geom_line(linewidth = 1, color = "darkblue") +
    labs(title = paste("AR(1) with rho =", rho), y = NULL)
  p_list <- c(p_list, list(p))
}

p_list[[1]] / p_list[[2]] / p_list[[3]]
```



As $|\rho| \to 1$, the time series loses its mean reversion.

Exercise 15.2: Cointegration vs. correlation

Consider the cointegration model of two time series with a common trend:

$$y_{1t} = x_t + w_{1t},$$
$$y_{2t} = x_t + w_{2t},$$

where x_t is a stochastic common trend defined as a random walk,

$$x_t = x_{t-1} + w_t,$$

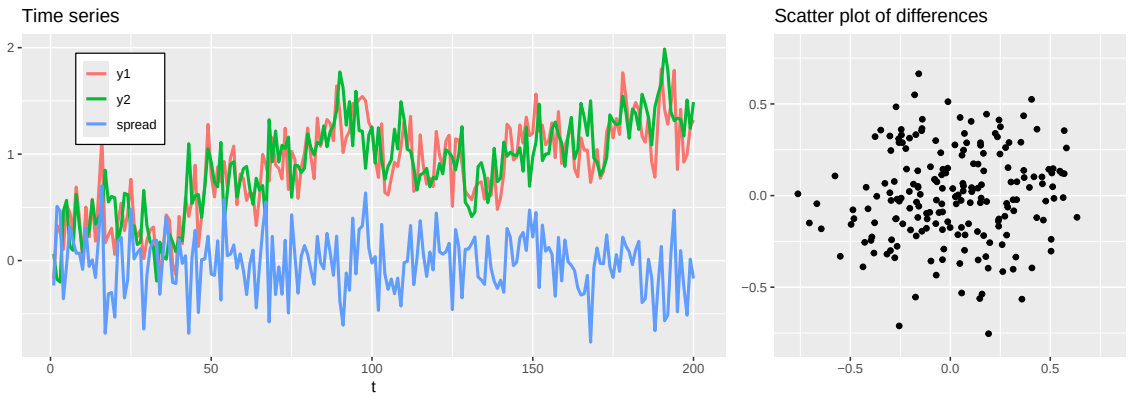
and the terms w_{1t}, w_{2t}, w_t are i.i.d. residual terms, mutually independent, with variances σ_1^2, σ_2^2, and σ^2, respectively.
Generate realizations of such time series with different values for the residual variances and plot the sequences as well as the scatter plot of the series differences (Δy_{1t} vs. Δy_{2t}). Choose the appropriate values of the variances to obtain cointegrated time series with low correlation as well as non-cointegrated time series with high correlation.

Solution

- Example of cointegrated time series with low correlation: We use $\sigma = 0.1$ and $\sigma_1 = \sigma_2 = 0.2$. The theoretical correlation is $\rho = 0.111$, whereas the empirical correlation computed with 200 observations is $\rho = 0.034$, and with 2 000 observations is $\rho = 0.108$.

```
# generate synthetic data
set.seed(421)
T <- 200
gamma <- 1
w <- rnorm(T, sd = 0.1)
x <- cumsum(w)
y1 <- gamma*x + rnorm(T, sd = 0.2)
y2 <-       x + rnorm(T, sd = 0.2)
spread <- y1 - y2
#cor(diff(y1), diff(y2))  # 0.03433501
```

```r
# plots
df <- rbind(data.frame("t"        = 1:T,
                       "series"   = "y1",
                       "value"    = y1),
            data.frame("t"        = 1:T,
                       "series"   = "y2",
                       "value"    = y2),
            data.frame("t"        = 1:T,
                       "series"   = "spread",
                       "value"    = y1 - y2))
df$series <- factor(df$series, levels = unique(df$series))

p_time_series <- ggplot(df, aes(x = t, y = value, color = series)) +
  geom_line(linewidth = 1) +
  theme(legend.title=element_blank(),
        legend.position = "inside",
        legend.position.inside = c(.14, .8),
        legend.background = element_rect(linewidth = 0.4, colour ="black")) +
  labs(title = "Time series", y = NULL)

p_scatter <- data.frame("log-return 1" = diff(y1), "log-return 2" = diff(y2),
                        check.names= FALSE) |>
  ggplot(aes(`log-return 1`, `log-return 2`)) +
  geom_point() +
  xlim(c(-0.8,0.8)) + ylim(c(-0.8,0.8)) +
  labs(title = "Scatter plot of differences", x = NULL, y = NULL)

p_time_series + p_scatter + plot_layout(ncol = 2, widths = c(2, 1))
```



The spread z_t is stationary and the scatter plot of the differences Δy_{1t} vs. Δy_{2t} does not show any preferred direction as expected for low correlation.

- Example of non-cointegrated time series with high correlation: We use $\sigma = 0.3$ and $\sigma_1 = \sigma_2 = 0.05$. The theoretical correlation is $\rho = 0.111$, whereas the empirical correlation computed with 200 observations is $\rho = 0.034$, and with 2 000 observations is $\rho = 0.108$. In addition, we add the linear trend $0.01 \times t$ to the first time series y_{1t}, which will destroy the cointegration between the two time series while not affecting the correlation. In this case, the theoretical correlation is $\rho = 0.947$, whereas the empirical correlation computed with 200 observations is $\rho = 0.952$, and with 2 000 observations $\rho = 0.941$.

```
# generate synthetic data
set.seed(421)
T <- 200
gamma <- 1
w <- rnorm(T, sd = 0.3)
x <- cumsum(w)
y1 <- gamma*x + rnorm(T, sd = 0.05) + 0.01*c(1:T - 1)
y2 <-       x + rnorm(T, sd = 0.05)
spread <- y1 - y2
#cor(diff(y1), diff(y2))  # 0.9521495
```

```r
# plots
df <- rbind(data.frame("t"        = 1:T,
                       "series"   = "y1",
                       "value"    = y1),
            data.frame("t"        = 1:T,
                       "series"   = "y2",
                       "value"    = y2),
            data.frame("t"        = 1:T,
                       "series"   = "spread",
                       "value"    = y1 - y2))
df$series <- factor(df$series, levels = unique(df$series))

p_time_series <- ggplot(df, aes(x = t, y = value, color = series)) +
  geom_line(linewidth = 1) +
  theme(legend.title=element_blank(),
        legend.position = "inside",
        legend.position.inside = c(.14, .8),
        legend.background = element_rect(linewidth = 0.4, colour ="black")) +
  labs(title = "Time series", y = NULL)

p_scatter <- data.frame("log-return 1" = diff(y1), "log-return 2" = diff(y2),
                        check.names= FALSE) |>
  ggplot(aes(`log-return 1`, `log-return 2`)) +
  geom_point() +
  xlim(c(-0.8,0.8)) + ylim(c(-0.8,0.8)) +
  labs(title = "Scatter plot of differences", x = NULL, y = NULL)

p_time_series + p_scatter + plot_layout(ncol = 2, widths = c(2, 1))
```

The spread z_t is nonstationary and the scatter plot of the differences Δy_{1t} vs. Δy_{2t} clearly shows a preferred direction as expected for high correlation.

Exercise 15.3: Simple pairs trading on AR(1) spread

Generate a synthetic mean-reverting spread with an AR(1) model for the log-prices, implement a simple pairs trading strategy based on thresholds, and plot the cumulative return (ignoring transaction costs).
Note: with a buy position, the portfolio return is the same as that of the spread; with a short position, it is the opposite; and with no position, it is just zero.

Solution

First, we generate the synthetic AR(1) signal:

```
# generate synthetic spread (log-prices) as an AR(1)
T <- 200
rho <- 0.7
z_mu <- 0.02
z_sd <- 0.10
phi <- (1 - rho)*z_mu
sd_eps <- sqrt(1 - rho^2)*z_sd

set.seed(42)
z <- rep(NA, T)
z[1] <- phi + rnorm(1, sd = sd_eps)
for (t in 2:T)
  z[t] <- phi + rho*z[t - 1] + rnorm(1, sd = sd_eps)
```

Then, we define the pairs trading strategy based on thresholds. The simplest implementation is based on comparing the z-score to a threshold s_0: buy when the z-score is below $-s_0$ and short-sell when it is above s_0, while unwinding the position after reverting to the equilibrium value of zero. In terms of the sizing signal:

$$s_t = \begin{cases} +1 & \text{if } z_t^{\text{score}} < -s_0, \\ 0 & \text{after } z_t^{\text{score}} \text{ reverts to } 0, \\ -1 & \text{if } z_t^{\text{score}} > +s_0. \end{cases}$$

```r
generate_thresholded_signal <- function(z_score, s0 = 1, start_signal_at = 1) {
  T <- NROW(z_score)
  signal <- z_score
  signal[] <- 0

  # initial point
  if (z_score[start_signal_at] < -s0) {  # buy
    signal[start_signal_at] <- 1
  } else if (z_score[start_signal_at] > s0)   # short-sell
      signal[start_signal_at] <- -1

  # loop for other points
  for (t in (start_signal_at+1):T) {
    if (z_score[t] < -s0)  # buy
      signal[t] <- 1
    else if (z_score[t] > s0)   # short-sell
      signal[t] <- -1
    else {  # maintain position or close position
      if (signal[t-1] == 1 && z_score[t] < 0)  # maintain buy position
        signal[t] <- signal[t-1]
      else if (signal[t-1] == -1 && z_score[t] > 0)  # maintain short-sell position
        signal[t] <- signal[t-1]
    }
  }

  return(signal)
}
```

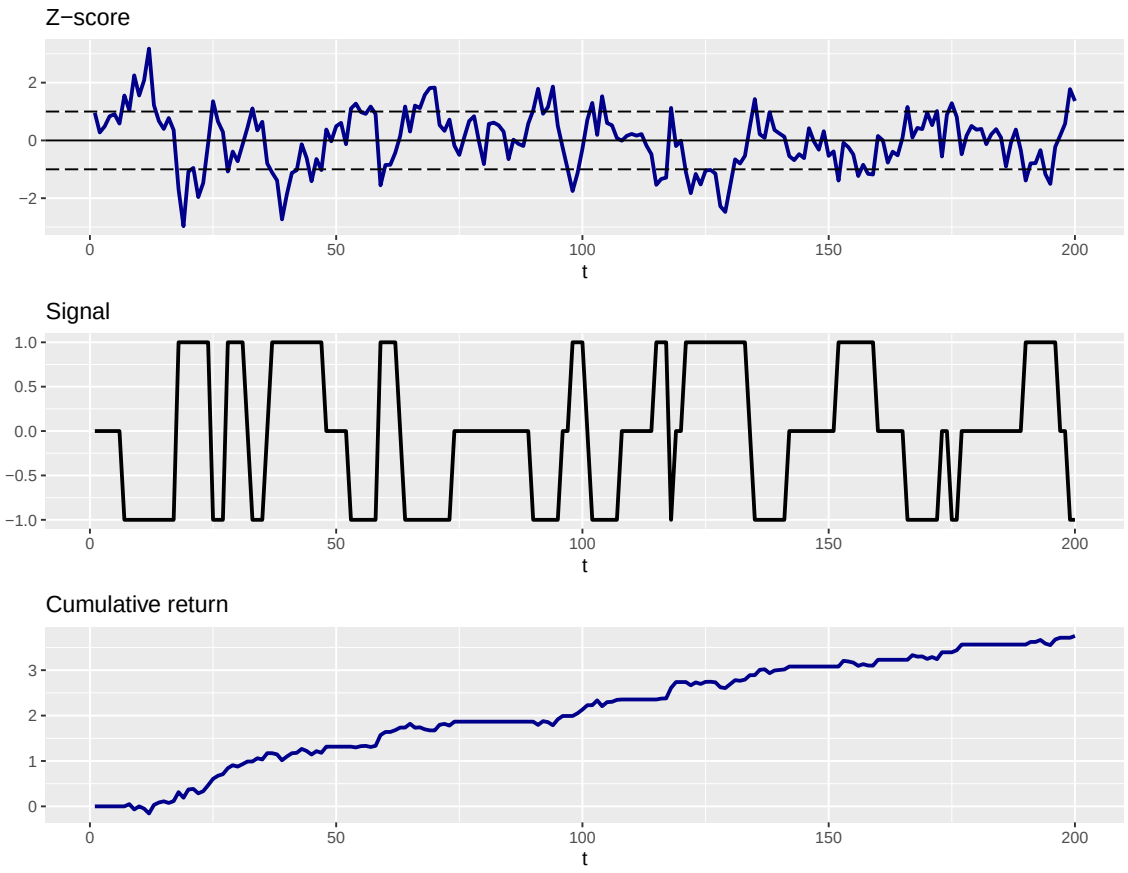
And we define the function that will backtest the strategy based on the signal generated:

```r
compute_cumPnL_spread_trading <- function(spread, signal, compounded = FALSE) {
  spread <- as.vector(spread)
  signal <- as.vector(signal)
  T <- length(spread)

  signal_delayed <- c(0, signal[-T])
  spread_ret <- c(0, diff(spread))
  portf_ret <- signal_delayed*spread_ret
  if (compounded)
    portf_cumret <- cumprod(1 + portf_ret) - 1
  else
    portf_cumret <- cumsum(portf_ret)

  return(portf_cumret)
}
```

Finally, we are ready to perform the backtest of pairs trading via the thresholded strategy on the spread:

```
# compute Z-score
z_score <- (z - mean(z))/sd(z)

# generate signal
s0 <- 1
signal <- generate_thresholded_signal(z_score, s0)

# compute P&L
portf_cumret <- compute_cumPnL_spread_trading(z, signal)
```

```
# plots
p_zscore <- ggplot(data.frame(t = 1:T, value = z_score), aes(x = t, y = value)) +
  geom_line(linewidth = 1, color = "darkblue") +
  geom_hline(yintercept = -s0, color = "black" , linetype = "longdash") +
  geom_hline(yintercept = 0,   color = "black" , linetype = "solid") +
  geom_hline(yintercept = s0,  color = "black" , linetype = "longdash") +
  labs(title = "Z-score", y = NULL)
p_signal <- ggplot(data.frame(t = 1:T, value = signal), aes(x = t, y = value)) +
  geom_line(linewidth = 1, color = "black") +
  labs(title = "Signal", y = NULL)
p_cumret <- ggplot(data.frame(t = 1:T, value = portf_cumret), aes(x = t, y = value)) +
  geom_line(linewidth = 1, color = "darkblue") +
  labs(title = "Cumulative return", y = NULL)

p_zscore / p_signal / p_cumret
```



Exercise 15.4: Discovering cointegrated pairs

a. Download market data corresponding to several assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Implement a prescreening approach on different pairs based on normalized prices.
c. Then consider running cointegration tests on the successful pairs from the prescreening phase. In particular, try some of the following tests:

- DF
- ADF
- PP
- PGFF
- ERSD

- JOT
- SPR

d. Plot the spreads of the successful cointegrated pairs as well as some of the unsuccessful ones for comparison.

Solution

a. We download market data for the ETFs EWA and EWC, as well as Coca-Cola and Pepsi:

```
library(xts)
library(quantmod)

# Download data
xts_pair_EWA_EWC <- cbind(Ad(getSymbols("EWA",
                                        from = "2016-01-01", to = "2019-12-31",
                                        auto.assign = FALSE)),
                          Ad(getSymbols("EWC",
                                        from = "2016-01-01", to = "2019-12-31",
                                        auto.assign = FALSE))
                          )
colnames(xts_pair_EWA_EWC) <- c("EWA", "EWC")

xts_pair_KO_PEP <- cbind(Ad(getSymbols("KO",
                                       from = "2017-01-01", to = "2019-12-31",
                                       auto.assign = FALSE)),
                         Ad(getSymbols("PEP",
                                       from = "2017-01-01", to = "2019-12-31",
                                       auto.assign = FALSE)))
colnames(xts_pair_KO_PEP) <- c("KO", "PEP")
```

b. Prescreening based on normalized price difference:

$$\text{NPD} \triangleq \sum_{t=1}^{T} (\tilde{p}_{1t} - \tilde{p}_{2t})^2 ,$$

where $\tilde{p}_{1t} = p_{1t}/p_{10}$ and $\tilde{p}_{2t} = p_{2t}/p_{20}$ are the normalized prices.

```r
# Assuming xts_pair is an xts object with two columns of price data
compute_npd <- function(xts_pair) {
  # Extract the two price series
  p1 <- xts_pair[, 1]
  p2 <- xts_pair[, 2]

  # Get initial prices (p_10 and p_20)
  p10 <- as.numeric(p1[1])
  p20 <- as.numeric(p2[1])

  # Normalize the prices
  p1_tilde <- p1 / p10
  p2_tilde <- p2 / p20

  # Compute the squared differences
  squared_diff <- (p1_tilde - p2_tilde)^2

  # Sum the squared differences to get NPD
  npd <- sum(squared_diff) / nrow(xts_pair)

  return(npd)
}

npd_pair_EWA_EWC <- compute_npd(xts_pair_EWA_EWC)
npd_pair_KO_PEP <- compute_npd(xts_pair_KO_PEP)

cat("Normalized price difference for EWA and EWC:", npd_pair_EWA_EWC, "\n")
```

```
Normalized price difference for EWA and EWC: 0.006858374
```

```r
cat("Normalized price difference for Coca-Cola and Pepsi:", npd_pair_KO_PEP, "\n")
```

```
Normalized price difference for Coca-Cola and Pepsi: 0.002156237
```

c. Cointegration tests:

- Cointegration tests for EWA vs. EWC:

```r
library(egcm)
cointegration_pair_EWA_EWC <- egcm(xts_pair_EWA_EWC[, 1], xts_pair_EWA_EWC[, 2])
summary(cointegration_pair_EWA_EWC)
```

```
EWC[i] =   1.2950 EWA[i] +   2.4541 + R[i], R[i] =   0.9638 R[i-1] + eps[i], eps ~ N(0,  0.1403^2)
          (0.0099)          (0.1734)                (0.0086)
```

```
R[2019-12-30] = 0.0223 (t = 0.043)

Unit Root Tests of Residuals
                                                  Statistic    p-value
  Augmented Dickey Fuller (ADF)                      -4.440    0.00486
  Phillips-Perron (PP)                              -38.361    0.00577
  Pantula, Gonzales-Farias and Fuller (PGFF)          0.965    0.00615
  Elliott, Rothenberg and Stock DF-GLS (ERSD)        -1.001    0.53105
  Johansen's Trace Test (JOT)                       -28.609    0.00688
  Schmidt and Phillips Rho (SPR)                    -11.987    0.38398

Variances
  SD(diff(EWA))        =   0.155061
  SD(diff(EWC))        =   0.191063
  SD(diff(residuals))  =   0.142003
  SD(residuals)        =   0.515301
  SD(innovations)      =   0.140338

Half life       =  18.785101
R[last]         =   0.022327 (t=0.04)
```

We can observe that when EWC is regressed against EWA, the hedge ratio is $\gamma = 1.2950$, and the residual has an AR coefficient of $\rho = 0.9638$, which seems far from a unit root $\rho = 1$ and is verified from the unit-root tests. Recall that the p-value is the probability that the observed data comes from the null hypothesis of a unit rood $\rho = 1$ present. Unders most of the tests, the p-value is small enough to reject the null hypothesis, meaning that it is mean reverting.

- Cointegration tests for Coca-Cola vs. Pepsi:

```
library(egcm)

cointegration_pair_KO_PEP <- egcm(xts_pair_KO_PEP[, 1], xts_pair_KO_PEP[, 2])
summary(cointegration_pair_KO_PEP)
```

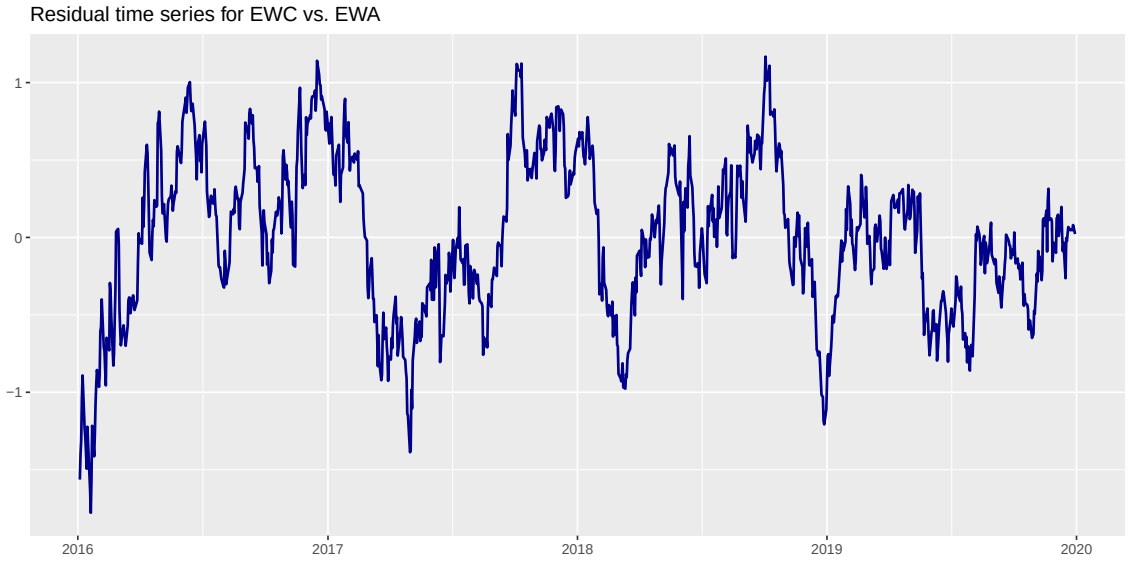
```
PEP[i] =   2.4108 KO[i] +   4.6813 + R[i], R[i] =   0.9901 R[i-1] + eps[i], eps ~ N(0,  0.7066^2)
          (0.0336)         (1.3320)                (0.0069)

R[2019-12-30] = -0.5081 (t = -0.136)

WARNING: KO and PEP do not appear to be cointegrated.

Unit Root Tests of Residuals
                                                  Statistic    p-value
  Augmented Dickey Fuller (ADF)                      -2.518    0.26936
  Phillips-Perron (PP)                              -14.136    0.18506
  Pantula, Gonzales-Farias and Fuller (PGFF)          0.981    0.13946
```

```
  Elliott, Rothenberg and Stock DF-GLS (ERSD)          -2.504    0.04857
  Johansen's Trace Test (JOT)                         -10.826    0.56213
  Schmidt and Phillips Rho (SPR)                      -16.657    0.20348

Variances
  SD(diff(KO))          =   0.349172
  SD(diff(PEP))         =   0.877866
  SD(diff(residuals))   =   0.709129
  SD(residuals)         =   3.723492
  SD(innovations)       =   0.706575

Half life       =  69.615843
R[last]         =  -0.508107 (t=-0.14)
```

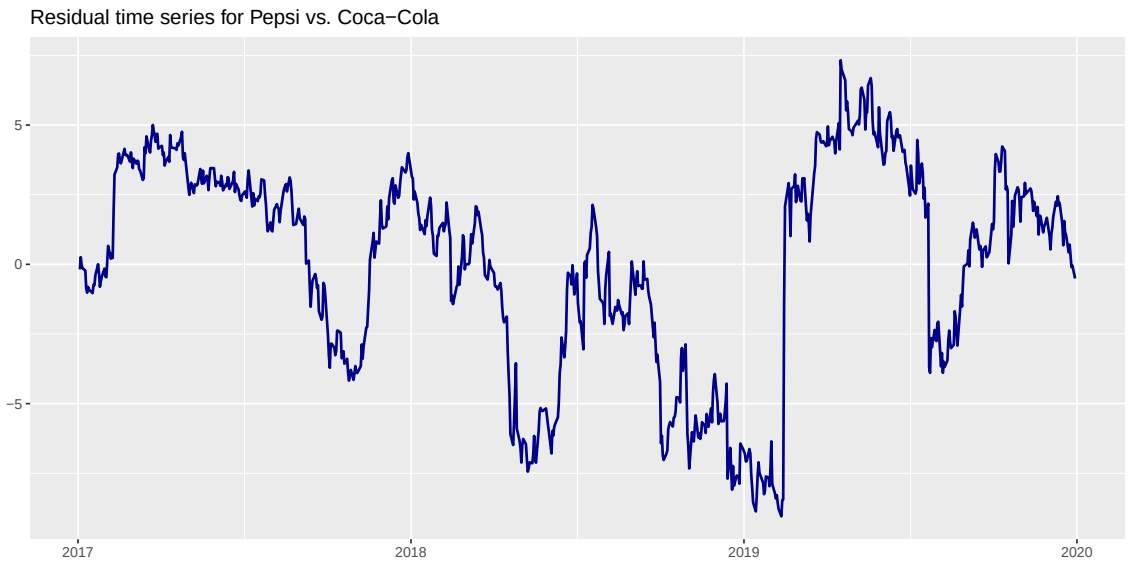
In this case, when Pepsi is regressed against Coca-Cola, the residual has an AR coefficient of $\rho = 0.99$, which is too close to a unit root $\rho = 1$ and may be an indication of not cointegration. Indeed, the p-values from all the tests are high and the null hypothesis cannot be rejected, meaning that we cannot conclude that it is mean reverting.

d. We now plot the spread or residual of the pairs:

```r
# plot residual
xts(cointegration_pair_EWA_EWC$residuals, index(xts_pair_EWA_EWC)) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Residual time series for EWC vs. EWA", x = NULL, y = NULL)
```



```r
# plot residual
xts(cointegration_pair_KO_PEP$residuals, index(xts_pair_KO_PEP)) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Residual time series for Pepsi vs. Coca-Cola", x = NULL, y = NULL)
```



Exercise 15.5: Pairs trading with least squares

a. Download market data corresponding to a pair of cointegrated assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Using an initial window as training data, estimate the hedge ratio γ via least squares.
c. Using that hedge ratio, compute the normalized spread (with leverage 1) in the remaining window as test data, that is, a spread obtained using the normalized portfolio

$$\boldsymbol{w} = \frac{1}{1+\gamma}\begin{bmatrix} 1 \\ -\gamma \end{bmatrix}.$$

d. Trade the normalized spread via the thresholded strategy.
e. Plot the cumulative return ignoring transaction costs.
f. Plot the cumulative return including transaction costs (e.g., as 30–90 bps of the portfolio turnover).

Solution

We will consider pairs trading on EWA–EWC with six-month rolling z-score and two-year fixed least squares.

a. We download market data for the ETFs EWA and EWC:

```
library(xts)
library(quantmod)

# download data
y1 <- log(Ad(getSymbols("EWA", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
y2 <- log(Ad(getSymbols("EWC", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
T <- nrow(y1)
T_trn <- 2*252
```

b. Using an initial window as training data, estimate the hedge ratio γ via least squares:

```
# hedge ratio via fixed LS
gamma <- lm(y1[1:T_trn] ~ y2[1:T_trn] + 1)$coefficients[2]
```

c. Using that hedge ratio, compute the normalized spread (with leverage 1) in the remaining window as test data, that is, a spread obtained using the normalized portfolio

$$\boldsymbol{w} = \frac{1}{1+\gamma}\begin{bmatrix} 1 \\ -\gamma \end{bmatrix}.$$

```
# compute spread with leverage = 1
w <- c(1, -gamma)/(1 + gamma)  # y1 ~ y2
spread <- xts(cbind(y1, y2) %*% w, index(y1))
spread <- spread - mean(spread[1:T_trn])
```

d. Trade the normalized spread via the thresholded strategy.

We define the function `generate_BB_thresholded_signal()` that computes the z-score adaptively (to avoide look-ahead bias):

```r
library(TTR)

generate_BB_thresholded_signal <- function(
    spread,
    entry_zscore = 1,
    exit_zscore = 0,
    lookback = 20,
    start_signal_at = lookback
    ) {
  # compute Z-score via Bollinger Bands
  bb <- BBands(spread, n = lookback, sd = 1)
  sdev <- (bb[, "up"] - bb[, "dn"])/2
  z_score <- (spread - bb[, "mavg"])/sdev

  # signal
  T <- NROW(spread)
  signal <- spread
  signal[] <- 0
  start_signal_at <- max(start_signal_at, lookback)
  for (t in start_signal_at:T) {
    if (z_score[t] < -entry_zscore)  # buy
      signal[t] <- 1
    else if (z_score[t] > entry_zscore)   # short-sell
      signal[t] <- -1
    else {  # maintain position or close position
      if (signal[t-1] == 1 && z_score[t] < -exit_zscore)  # maintain buy position
        signal[t] <- signal[t-1]
      else if (signal[t-1] == -1 && z_score[t] > exit_zscore)  # maintain short-sell position
        signal[t] <- signal[t-1]
    }
  }

  return(list(signal = signal, z_score = z_score))
}
```

We now generate the signal:

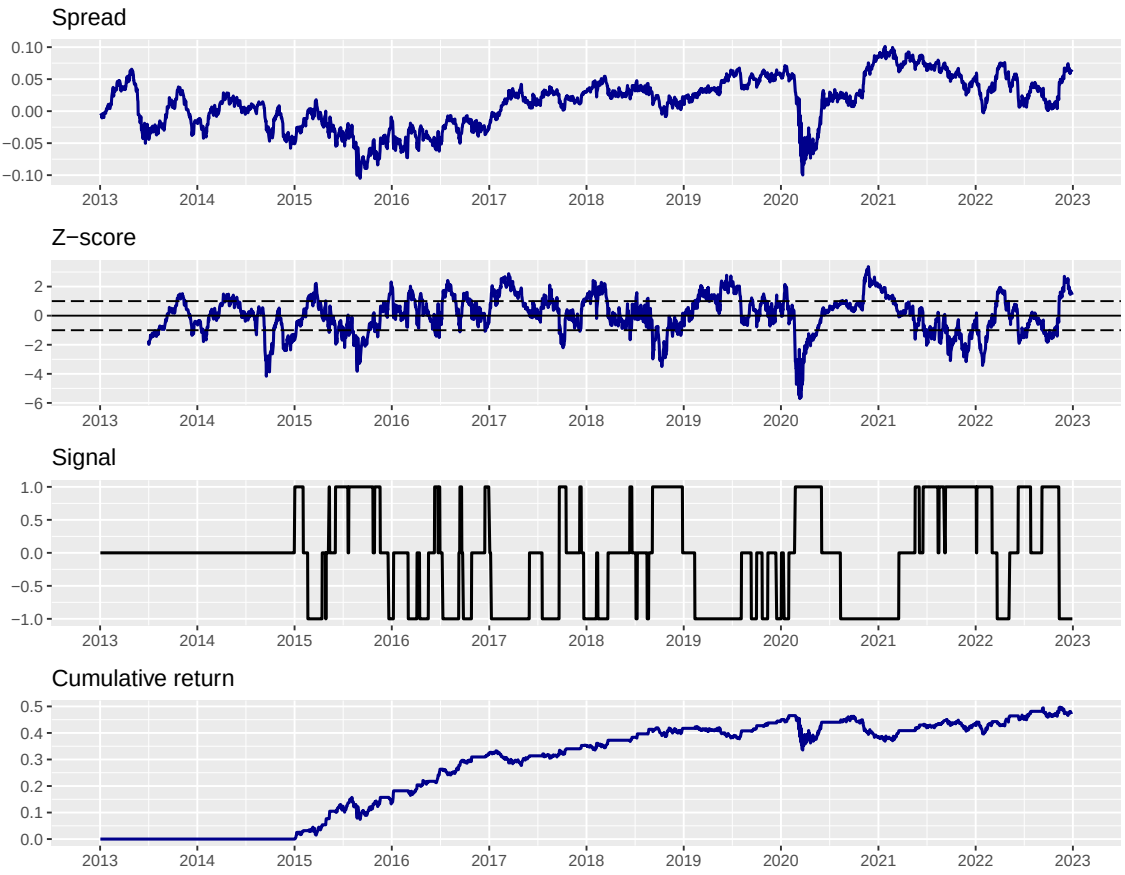
```r
# generate sizing signal via thresholded strategy
entry_zscore = 1
res_BB <- generate_BB_thresholded_signal(spread, entry_zscore,
                                         lookback = 6*21,
                                         start_signal_at = T_trn+1)
z_score <- res_BB$z_score
signal  <- res_BB$signal
```

And then define the function `compute_cumPnL_spread_trading()` to perform the backtest:

```r
compute_cumPnL_spread_trading <- function(spread, signal, compounded = FALSE) {
  if (!all.equal(index(spread), index(signal)))
    stop("Temporal indices not aligned in spread and signal.")
  if (is.xts(spread))
    index_orig <- index(spread)
  spread <- as.vector(spread)
  signal <- as.vector(signal)
  T <- length(spread)

  signal_delayed <- c(0, signal[-T])
  spread_ret <- c(0, diff(spread))
  portf_ret <- signal_delayed*spread_ret
  if (compounded)
    portf_cumret <- cumprod(1 + portf_ret) - 1
  else
    portf_cumret <- cumsum(portf_ret)

  if (exists("index_orig"))
    portf_cumret <- xts(portf_cumret, index_orig)
  return(portf_cumret)
}

# compute P&L
portf_cumret <- compute_cumPnL_spread_trading(spread, signal)
```

e. Plot the cumulative return ignoring transaction costs.

```r
# plots
p_spread <- fortify(spread, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Spread", x = NULL, y = NULL)

p_zscore <- fortify(z_score, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  geom_hline(yintercept = -entry_zscore, color = "black" , linetype = 5) +
  geom_hline(yintercept = 0, color = "black" , linetype = 1) +
  geom_hline(yintercept = entry_zscore, color = "black" , linetype = 5) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Z-score", x = NULL, y = NULL)

p_signal <- fortify(signal, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "black") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Signal", x = NULL, y = NULL)

p_cumret <- fortify(portf_cumret, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative return", x = NULL, y = NULL)

p_spread / p_zscore / p_signal / p_cumret
```



f. Plot the cumulative return including transaction costs (e.g., as 30–90 bps of the portfolio turnover).

We leave this minor modification of the function `compute_cumPnL_spread_trading()` to the user.

Exercise 15.6: Pairs trading with rolling least squares

Repeat Exercise 15.5 but using rolling least squares to track the hedge ratio over time γ_t.

Solution

We will consider pairs trading on EWA–EWC with six-month rolling z-score and two-year rolling least squares.

a. We download market data for the ETFs EWA and EWC:

```
library(xts)
library(quantmod)

# download data
y1 <- log(Ad(getSymbols("EWA", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
y2 <- log(Ad(getSymbols("EWC", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
T <- nrow(y1)
T_trn <- 2*252
```

b. Estimate the hedge ratio γ via least squares on a rolling-window basis:

```
fit_rollingLS <- function(y1, y2, lookback = 2*252, every = 50) {
  T <- NROW(y1)
  # create update points
  t_update <- seq(from = lookback, to = T, by = every)
  if (last(t_update) < T)
    t_update <- c(t_update, T)
  # LS loop
  gamma <- mu <- y1
  gamma[] <- mu[] <- NA
  for (t in t_update) {
    res_LS <- lm(y1[(t-lookback+1):t] ~ y2[(t-lookback+1):t] + 1)
    gamma[t] <- res_LS$coefficients[2]
    mu[t]    <- res_LS$coefficients[1]
  }
  gamma <- na.locf(gamma)
  mu    <- na.locf(mu)
  # for make-up purposes (I will not trade during the first lookback samples anyway)
  gamma <- na.locf(gamma, fromLast = TRUE)
  mu    <- na.locf(mu, fromLast = TRUE)

  return(list(gamma = gamma, mu = mu))
}


# hedge ratio via rolling LS
rollingLS <- fit_rollingLS(y1, y2, lookback = T_trn, every = 1)
```

c. Using that hedge ratio, compute the normalized spread (with leverage 1) in the remaining window as test data, that is, a spread obtained using the normalized portfolio

$$\boldsymbol{w} = \frac{1}{1+\gamma}\begin{bmatrix} 1 \\ -\gamma \end{bmatrix}.$$

```
# compute spread with leverage=1
rollingLS$w <- cbind(1, -rollingLS$gamma)/as.numeric(1 + rollingLS$gamma)
rollingLS$spread <- xts(
  rowSums(cbind(y1, y2) * rollingLS$w) - rollingLS$mu/(1 + rollingLS$gamma),
  index(y1)
  )
```

d. Trade the normalized spread via the thresholded strategy.

We use the function `generate_BB_thresholded_signal()`, defined in Exercise 15.6, that computes the z-score adaptively (to avoide look-ahead bias) to generate the signal:

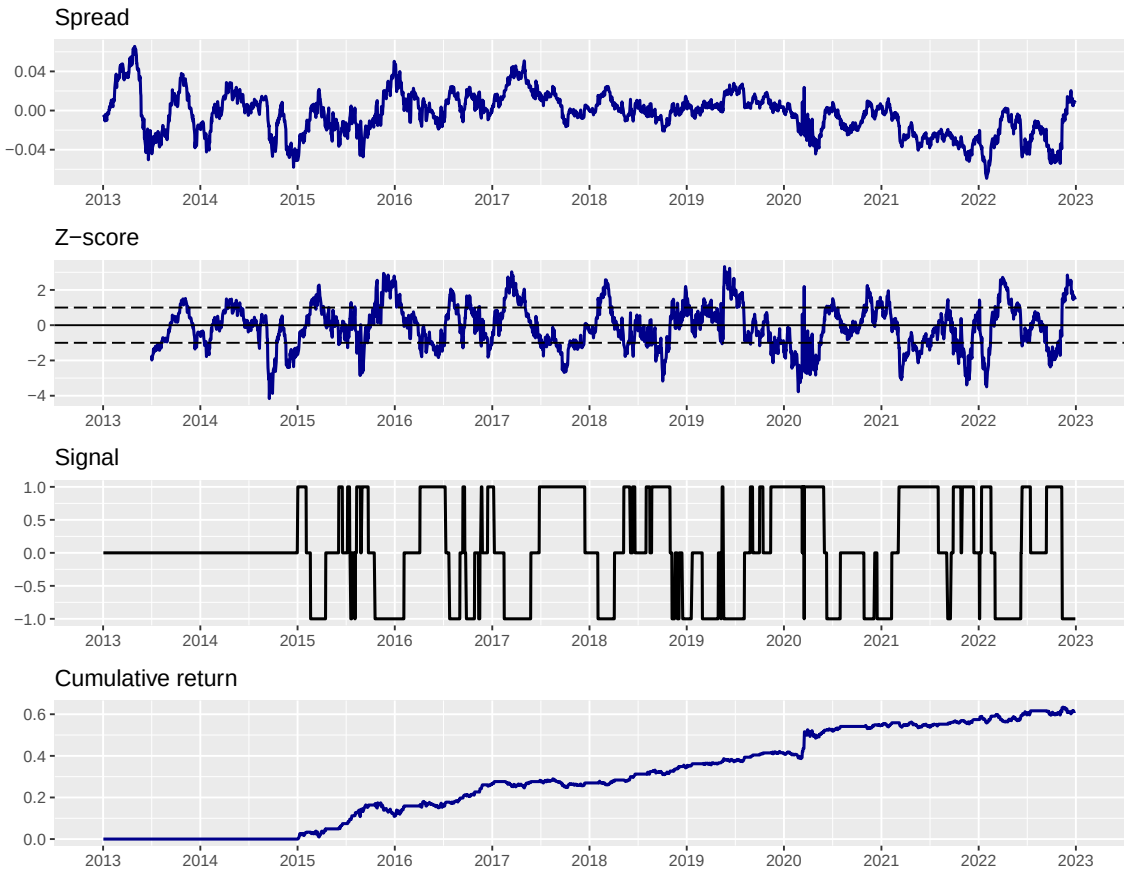
```
# generate sizing signal via thresholded strategy
entry_zscore = 1
rollingLS <- c(rollingLS,
               generate_BB_thresholded_signal(rollingLS$spread,
                                              entry_zscore,
                                              lookback = 6*21,
                                              start_signal_at = T_trn+1)
               )
```

And then use the function `compute_cumPnL_spread_trading()`, defined in Exercise 15.6, to perform the backtest:

```
# compute P&L
rollingLS$portf_cumret <- compute_cumPnL_spread_trading(rollingLS$spread,
                                                        rollingLS$signal)
```

e. Plot the cumulative return ignoring transaction costs.

```r
# plots
p_spread <- fortify(rollingLS$spread, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Spread", x = NULL, y = NULL)

p_zscore <- fortify(rollingLS$z_score, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  geom_hline(yintercept = -entry_zscore, color = "black" , linetype = 5) +
  geom_hline(yintercept = 0, color = "black" , linetype = 1) +
  geom_hline(yintercept = entry_zscore, color = "black" , linetype = 5) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Z-score", x = NULL, y = NULL)

p_signal <- fortify(rollingLS$signal, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "black") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Signal", x = NULL, y = NULL)

p_cumret <- fortify(rollingLS$portf_cumret, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative return", x = NULL, y = NULL)

p_spread / p_zscore / p_signal / p_cumret
```



Exercise 15.7: Pairs trading with Kalman filtering

Repeat Exercise 15.5 but using the Kalman filter to better track the hedge ratio over time γ_t.

Solution

We will consider pairs trading on EWA–EWC with six-month rolling z-score and Kalman for the hedge ratio tracking.

a. We download market data for the ETFs EWA and EWC:

```r
library(xts)
library(quantmod)

# download data
y1 <- log(Ad(getSymbols("EWA", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
y2 <- log(Ad(getSymbols("EWC", from = "2013-01-01", to = "2022-12-31", auto.assign = FALSE)))
T <- nrow(y1)
T_trn <- 2*252
```

b. Estimate the hedge ratio γ via Kalman (using the R package `MARSS`):

```
library(MARSS)

# Hedge ratio via Kalman
#
# Documentation:
#   RShowDoc("Quick_Start", package = "MARSS")
#   RShowDoc("UserGuide", package = "MARSS")
#

# initial LS
yc1 <- as.vector(y1[1:T_trn] - mean(y1[1:T_trn]))
yc2 <- as.vector(y2[1:T_trn] - mean(y2[1:T_trn]))
gamma_hat <- as.numeric((yc2 %*% yc1) / (yc2 %*% yc2))
mu_hat <- mean(y1[1:T_trn] - gamma_hat * y2[1:T_trn])
epsilon_hat <- y1[1:T_trn] - mu_hat - gamma_hat * y2[1:T_trn]
# Kalman parameters
var_eps <- as.numeric(var(epsilon_hat))
var_y2  <- as.numeric(var(y2[1:T_trn]))
mu1 <- mu_hat
var_mu1 <- (1/T_trn)*var_eps
gamma1 <- gamma_hat
var_gamma1 <- (1/T_trn)*var_eps/var_y2
# Kalman model #1 (basic)
alpha <- 1e-5
Zt <- array(list(), dim = c(1, 2, length(y2)))
Zt[1, 1, ] <- 1
Zt[1, 2, ] <- y2
model <- MARSS(rbind(as.vector(y1)),
               model = list(Z = Zt,  A = "zero", R = matrix(var_eps),  # observation equation
                            B = "identity",  U = "zero",  # state transition equation
                            Q = diag(alpha * c(var_eps, var_eps/var_y2)),
                            x0 = cbind(c(mu1, gamma1)),         # initial state
                            V0 = diag(c(var_mu1, var_gamma1))), # initial state
               fit = FALSE, silent = TRUE)
kfks <- MARSSkfss(model)
kalman_basic <- list()
kalman_basic$mu    <- xts(kfks$xtt1[1, ], index(y1))
kalman_basic$gamma <- xts(kfks$xtt1[2, ], index(y1))
```

c. Using that hedge ratio, compute the normalized spread (with leverage 1) in the remaining window as test data, that is, a spread obtained using the normalized portfolio

$$\boldsymbol{w} = \frac{1}{1+\gamma}\begin{bmatrix} 1 \\ -\gamma \end{bmatrix}.$$

```
# compute spread with leverage=1
kalman_basic$w <- cbind(1, -kalman_basic$gamma)/as.numeric(1 + kalman_basic$gamma)
kalman_basic$spread <- xts(
  rowSums(cbind(y1, y2) * kalman_basic$w) - kalman_basic$mu/(1 + kalman_basic$gamma),
  index(y1)
  )
```

d. Trade the normalized spread via the thresholded strategy.

We use the function `generate_BB_thresholded_signal()`, defined in Exercise 15.6, that computes the z-score adaptively (to avoide look-ahead bias) to generate the signal:

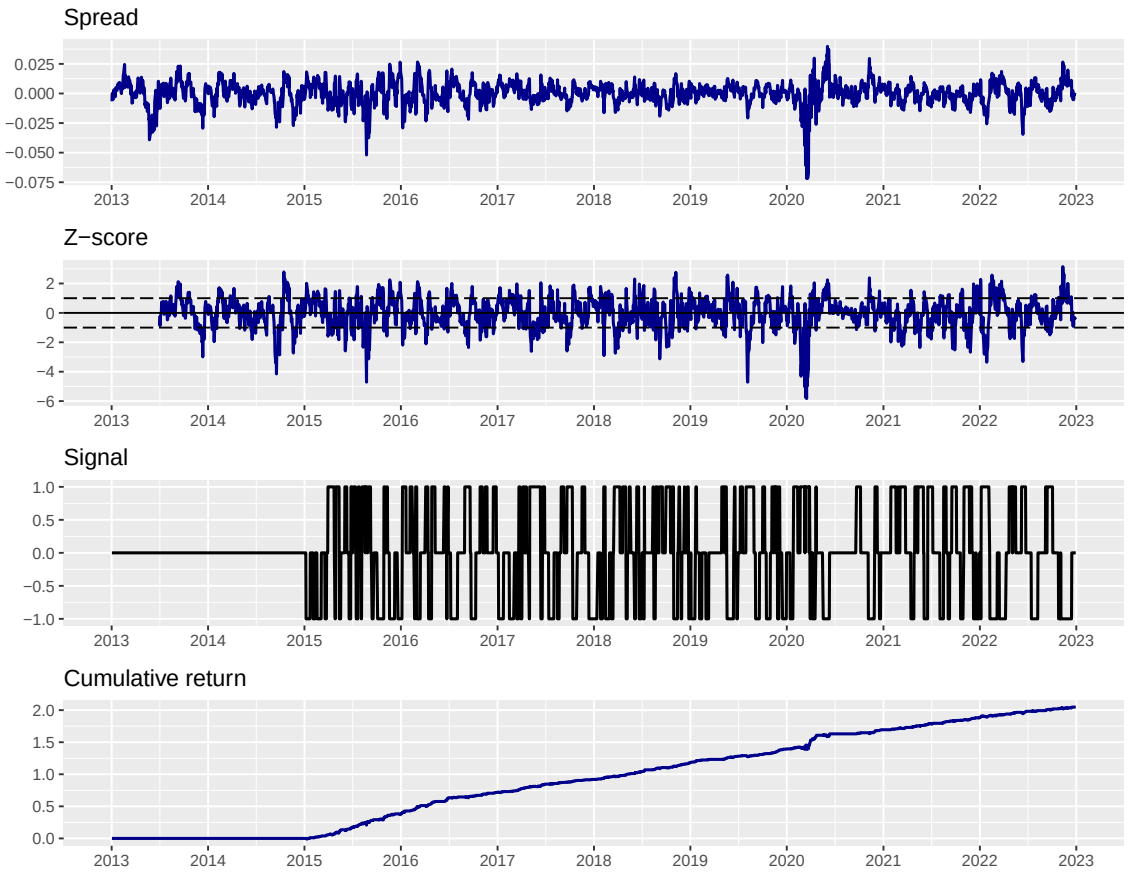
```
# generate sizing signal via thresholded strategy
entry_zscore = 1
kalman_basic <- c(kalman_basic,
                  generate_BB_thresholded_signal(kalman_basic$spread,
                                                 entry_zscore,
                                                 lookback = 6*21,
                                                 start_signal_at = T_trn+1)
                  )
```

And then use the function `compute_cumPnL_spread_trading()`, defined in Exercise 15.6, to perform the backtest:

```
# compute P&L
kalman_basic$portf_cumret <- compute_cumPnL_spread_trading(kalman_basic$spread,
                                                           kalman_basic$signal)
```

e. Plot the cumulative return ignoring transaction costs.

```r
# plots
p_spread <- fortify(kalman_basic$spread, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Spread", x = NULL, y = NULL)

p_zscore <- fortify(kalman_basic$z_score, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  geom_hline(yintercept = -entry_zscore, color = "black" , linetype = 5) +
  geom_hline(yintercept = 0, color = "black" , linetype = 1) +
  geom_hline(yintercept = entry_zscore, color = "black" , linetype = 5) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Z-score", x = NULL, y = NULL)

p_signal <- fortify(kalman_basic$signal, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "black") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Signal", x = NULL, y = NULL)

p_cumret <- fortify(kalman_basic$portf_cumret, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative return", x = NULL, y = NULL)

p_spread / p_zscore / p_signal / p_cumret
```



Exercise 15.8: Statistical arbitrage with more than two assets

a. Download market data corresponding to $N > 2$ cointegrated assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Choose a pair of assets and implement pairs trading via least squares.
c. With all the N assets, use VECM to obtain $K > 2$ cointegration relationships and then:
   - implement pairs trading with the strongest direction;
   - implement K parallel pairs trading and combine the result into a final cumulative return plot.
d. Compare and discuss the three implementations: pairs trading on just two assets, pairs trading on the strongest of the K directions, and K parallel pairs trading.

Solution

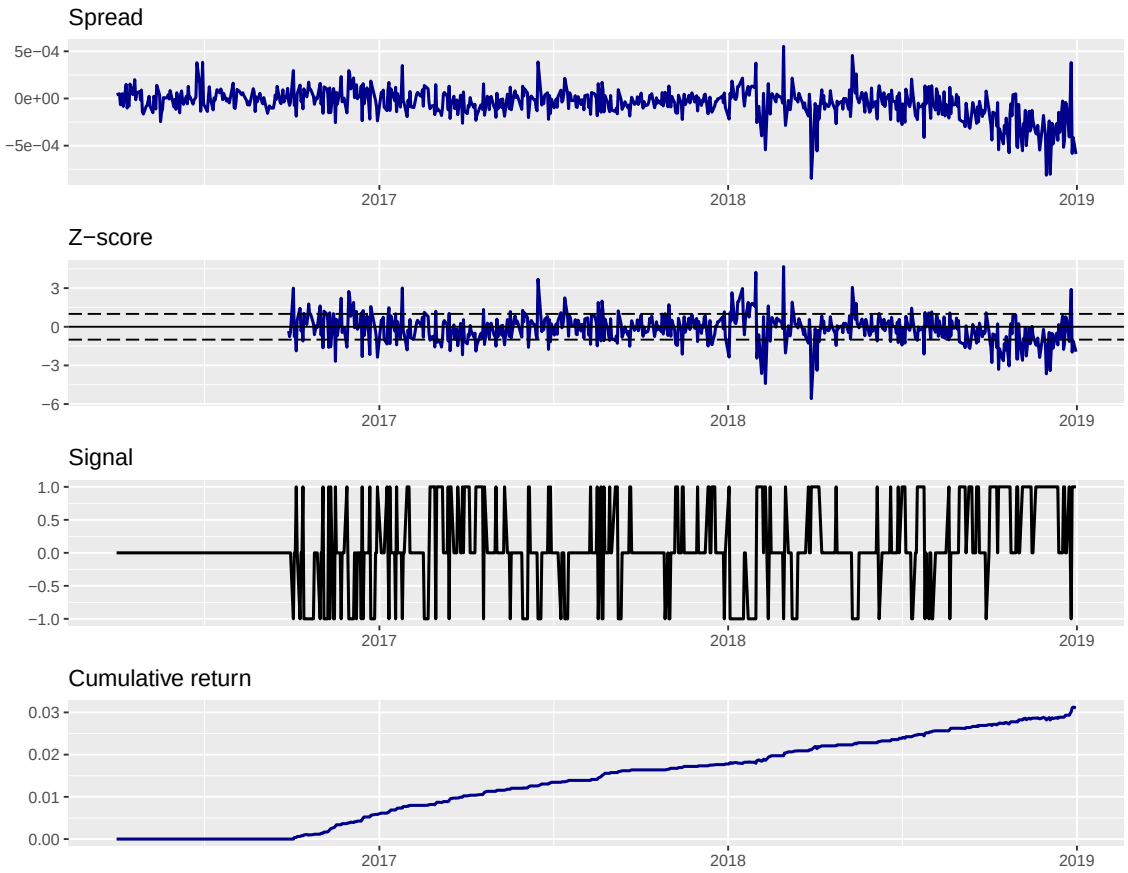
a. We download market data for three ETFs that track the S&P 500 index: Standard & Poor's Depository Receipts SPY, iShares IVV, and Vanguard's VOO:

```
library(xts)
library(quantmod)

# download data
xts_triplet <- cbind(Ad(getSymbols("SPY", from = "2013-01-01", to = "2022-12-31", auto.assign = FAL
                    Ad(getSymbols("IVV", from = "2013-01-01", to = "2022-12-31", auto.assign = FAL
                    Ad(getSymbols("VOO", from = "2013-01-01", to = "2022-12-31", auto.assign = FAL
xts_triplet <- log(xts_triplet)
colnames(xts_triplet) <- c("SPY", "IVV", "VOO")
xts_triplet <- xts_triplet["2016-04-01::2018-12-31", ]
T <- nrow(xts_triplet)
T_trn <- 6*21  #T_trn <- 2*252
```

b. Choose a pair of assets and implement pairs trading via least squares:

```
# hedge ratio via fixed LS
gamma <- lm(xts_triplet[1:T_trn, 1] ~ xts_triplet[1:T_trn, 2] + 1)$coefficients[2]

# compute spread with leverage = 1
w <- c(1, -gamma)/(1 + gamma)  # y1 ~ y2
spread <- xts(xts_triplet[, 1:2] %*% w, index(xts_triplet))
spread <- spread - mean(spread[1:T_trn])

# generate sizing signal via thresholded strategy
entry_zscore = 1
res_BB <- generate_BB_thresholded_signal(spread, entry_zscore,
                                         lookback = 6*21,
                                         start_signal_at = T_trn+1)
z_score <- res_BB$z_score
signal  <- res_BB$signal

# compute P&L
portf_cumret <- compute_cumPnL_spread_trading(spread, signal)
```

```r
# plots
p_spread <- fortify(spread, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Spread", x = NULL, y = NULL)

p_zscore <- fortify(z_score, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  geom_hline(yintercept = -entry_zscore, color = "black" , linetype = 5) +
  geom_hline(yintercept = 0, color = "black" , linetype = 1) +
  geom_hline(yintercept = entry_zscore, color = "black" , linetype = 5) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Z-score", x = NULL, y = NULL)

p_signal <- fortify(signal, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "black") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Signal", x = NULL, y = NULL)

p_cumret <- fortify(portf_cumret, melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative return", x = NULL, y = NULL)

p_spread / p_zscore / p_signal / p_cumret
```



c. With all the N assets, use VECM to obtain K cointegration relationships and then:

- implement pairs trading with the strongest direction;
- implement K parallel pairs trading and combine the result into a final cumulative return plot.

We detect $K = 2$ cointegration relationships and perform pairs trading on both:

```
library(urca)

# Johansen's cointegration test via VECM
res_urca <- urca::ca.jo(as.matrix(xts_triplet[1:T_trn, ]), type = "trace")
summary(res_urca)
```

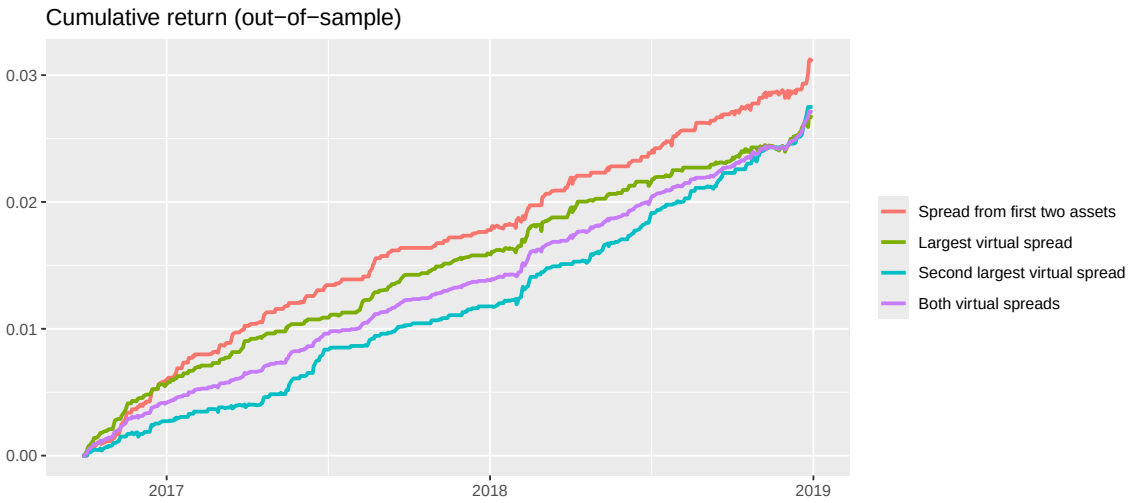
```
######################
# Johansen-Procedure #
######################
```

```
Test type: trace statistic , with linear trend

Eigenvalues (lambda):
[1] 0.37033518 0.35635073 0.01856731

Values of teststatistic and critical values of test:

           test 10pct  5pct  1pct
r <= 2 |   2.32  6.50  8.18 11.65
r <= 1 |  56.96 15.66 17.95 23.52
r = 0  | 114.32 28.71 31.52 37.22

Eigenvectors, normalised to first column:
(These are the cointegration relations)

          SPY.l2    IVV.l2    VOO.l2
SPY.l2  1.000000  1.000000  1.000000
IVV.l2 -0.235509 -9.349331 -5.512568
VOO.l2 -0.760224  8.343825 18.498382

Weights W:
(This is the loading matrix)

         SPY.l2    IVV.l2       VOO.l2
SPY.d -11.54811 0.7251951 -0.002634225
IVV.d -10.74623 0.8134947 -0.002645015
VOO.d -10.54496 0.7143952 -0.002669443
```

```
VECM1 <- list()
VECM2 <- list()
VECM1$beta <- as.vector(res_urca@V[, 1])
VECM2$beta <- as.vector(res_urca@V[, 2])
VECM1$w <- VECM1$beta/sum(abs(VECM1$beta))
VECM2$w <- VECM2$beta/sum(abs(VECM2$beta))

# compute spread with leverage = 1
VECM1$spread <- xts(xts_triplet %*% VECM1$w, index(xts_triplet))
VECM1$spread <- VECM1$spread - mean(VECM1$spread[1:T_trn])
VECM2$spread <- xts(xts_triplet %*% VECM2$w, index(xts_triplet))
VECM2$spread <- VECM2$spread - mean(VECM2$spread[1:T_trn])

# generate sizing signal via thresholded strategy
VECM1 <- c(VECM1,
           generate_BB_thresholded_signal(VECM1$spread,
                                          entry_zscore = 1,
                                          lookback = 6*21,
                                          start_signal_at = T_trn+1)
           )
VECM2 <- c(VECM2,
           generate_BB_thresholded_signal(VECM2$spread,
                                          entry_zscore = 1,
                                          lookback = 6*21,
                                          start_signal_at = T_trn+1)
           )

# compute P&L
VECM1$portf_cumret <- compute_cumPnL_spread_trading(VECM1$spread, VECM1$signal)
VECM2$portf_cumret <- compute_cumPnL_spread_trading(VECM2$spread, VECM2$signal)
```

Cumulative return for statistical arbitrage on SPY–IVV–VOO:

```
cumret_all <- cbind(portf_cumret,
                    VECM1$portf_cumret,
                    VECM2$portf_cumret,
                    0.5*(VECM1$portf_cumret + VECM2$portf_cumret))
colnames(cumret_all) <- c("Spread from first two assets",
                          "Largest virtual spread",
                          "Second largest virtual spread",
                          "Both virtual spreads")

fortify(cumret_all[c(T_trn:T), ], melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  theme(legend.title=element_blank()) +
  labs(title = "Cumulative return (out-of-sample)", x = NULL, y = NULL)
```



d. Compare and discuss the three implementations:

The previous cumulative return plot may be misleading and it is better to compare based on the Sharpe ratios:

```
library(PerformanceAnalytics)

t(SharpeRatio.annualized(diff(cumret_all[c(T_trn:T), ]),  geometric = FALSE))
```

```
                           Annualized Sharpe Ratio (Rf=0%)
Spread from first two assets                       5.841368
Largest virtual spread                             5.947125
```

```
Second largest virtual spread                    6.544023
Both virtual spreads                             8.704443
```

From the Sharpe ratios, we can observe the following:

- creating a spread based on just two assets is worse than using a virtual spread based on three assets,
- combining the two largest virtual spreads significantly improves using a single one (diversification),
- we would have expected the largest virtual spread to be better than the second one, but there is a small difference in the opposite direction which is probably not statistically significant.