# R_code

2023-04-30

## DATA Processing

### Read in data

### Convert data type

The `getSymbols()` function from the `quantmod` package returns a time series object of class `xts`. The dates are stored in the index of the object.

```
TE$Date <- as.Date(TE$Date, format = "%m/%d/%Y")
TE <- TE[order(TE$Date),]
TE <- TE[TE$Date>= as.Date("2012-01-01") & TE$Date<= as.Date("2022-12-31"),]
TE <- as.xts(TE)
```

### Summary statistics

```
logEX <- log(DEXTAUS)
dlogEX <- diff(logEX, lag=1, differences=1)
dlogEX <- na.omit(dlogEX)
summary(dlogEX)
```

```
##       Index                DEXTAUS
##  Min.   :2000-01-04   Min.   :-3.423e-02
##  1st Qu.:2005-09-25   1st Qu.:-1.274e-03
##  Median :2011-06-21   Median : 0.000e+00
##  Mean   :2011-06-25   Mean   :-3.630e-06
##  3rd Qu.:2017-03-23   3rd Qu.: 1.298e-03
##  Max.   :2022-12-30   Max.   : 3.200e-02
```

```
mean(dlogEX)
```

```
## [1] -3.631397e-06
```

```
sd(dlogEX)
```

```
## [1] 0.003038933
```

```
length(dlogEX)
```

```
## [1] 5764
```

```
logGSPC <- log(GSPC)
dlogGSPC <- diff(logGSPC, lag=1, differences=1)
dlogGSPC <- na.omit(dlogGSPC$GSPC.Close)
summary(dlogGSPC$GSPC.Close)
```

```
##       Index               GSPC.Close
##  Min.   :2000-01-04   Min.   :-0.1276522
```

```
##  1st Qu.:2005-10-05   1st Qu.:-0.0048987
##  Median :2011-07-05   Median : 0.0005737
##  Mean   :2011-07-05   Mean   : 0.0001677
##  3rd Qu.:2017-04-04   3rd Qu.: 0.0059151
##  Max.   :2022-12-30   Max.   : 0.1095720
```

```
mean(dlogGSPC$GSPC.Close)
```

```
## [1] 0.000167707
```

```
sd(dlogGSPC$GSPC.Close)
```

```
## [1] 0.01252844
```

```
length(dlogGSPC$GSPC.Close)
```

```
## [1] 5785
```

```
logTWII <- log(TWII)
dlogTWII <- diff(logTWII, lag=1, differences=1)
dlogTWII <- na.omit(dlogTWII$TWII.Close)
summary(dlogTWII$TWII.Close)
```

```
##       Index              TWII.Close
##  Min.   :2000-01-05   Min.   :-9.936e-02
##  1st Qu.:2005-09-23   1st Qu.:-5.798e-03
##  Median :2011-06-16   Median : 5.267e-04
##  Mean   :2011-06-25   Mean   : 8.474e-05
##  3rd Qu.:2017-03-23   3rd Qu.: 6.755e-03
##  Max.   :2022-12-30   Max.   : 6.525e-02
```

```
mean(dlogTWII$TWII.Close)
```

```
## [1] 8.474126e-05
```

```
sd(dlogTWII$TWII.Close)
```

```
## [1] 0.01320181
```

```
length(dlogTWII$TWII.Close)
```

```
## [1] 5653
```

```
logTE <- log(TE)
dlogTE <- diff(logTE, lag = 1, differences = 1)
dlogTE <- na.omit(dlogTE)
summary(dlogTE)
```

```
##       Index                dlogTE
##  Min.   :2012-01-03   Min.   :-0.0686833
##  1st Qu.:2014-09-28   1st Qu.:-0.0055024
##  Median :2017-06-29   Median : 0.0007009
##  Mean   :2017-06-30   Mean   : 0.0003390
##  3rd Qu.:2020-04-02   3rd Qu.: 0.0065642
##  Max.   :2022-12-30   Max.   : 0.0678243
```

```
mean(dlogTE)
```

```
## [1] 0.0003390508
```

```
sd(dlogTE)
```

```
## [1] 0.01110057
```

```
length(dlogTE)
```

```
## [1] 2708
```

## Unit root test

```
library(tseries)
adf.test(dlogEX) # ADF test
```

```
## Warning in adf.test(dlogEX): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dlogEX
## Dickey-Fuller = -16.216, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(dlogTE)
```

```
## Warning in adf.test(dlogTE): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dlogTE
## Dickey-Fuller = -13.272, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(dlogTWII)
```

```
## Warning in adf.test(dlogTWII): p-value smaller than printed p-value
```

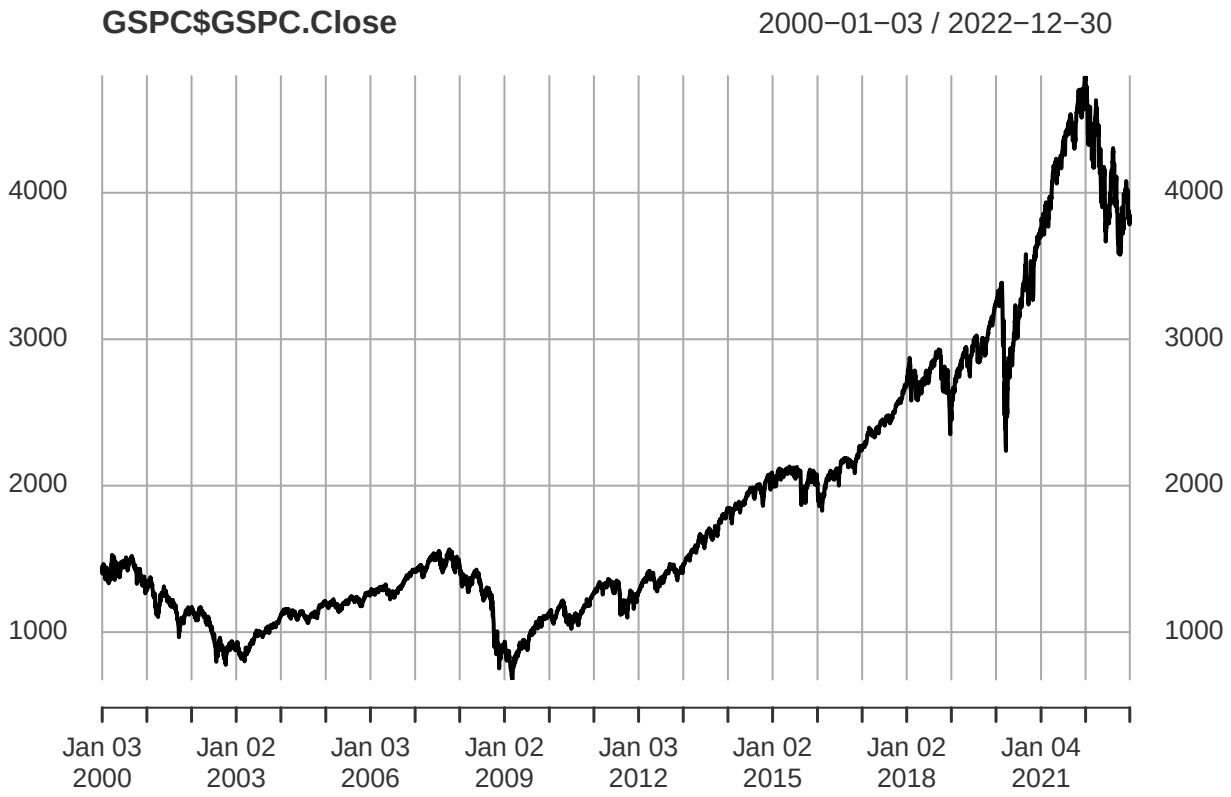
```
##
##  Augmented Dickey-Fuller Test
##
## data:  dlogTWII
## Dickey-Fuller = -16.768, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(dlogGSPC)
```

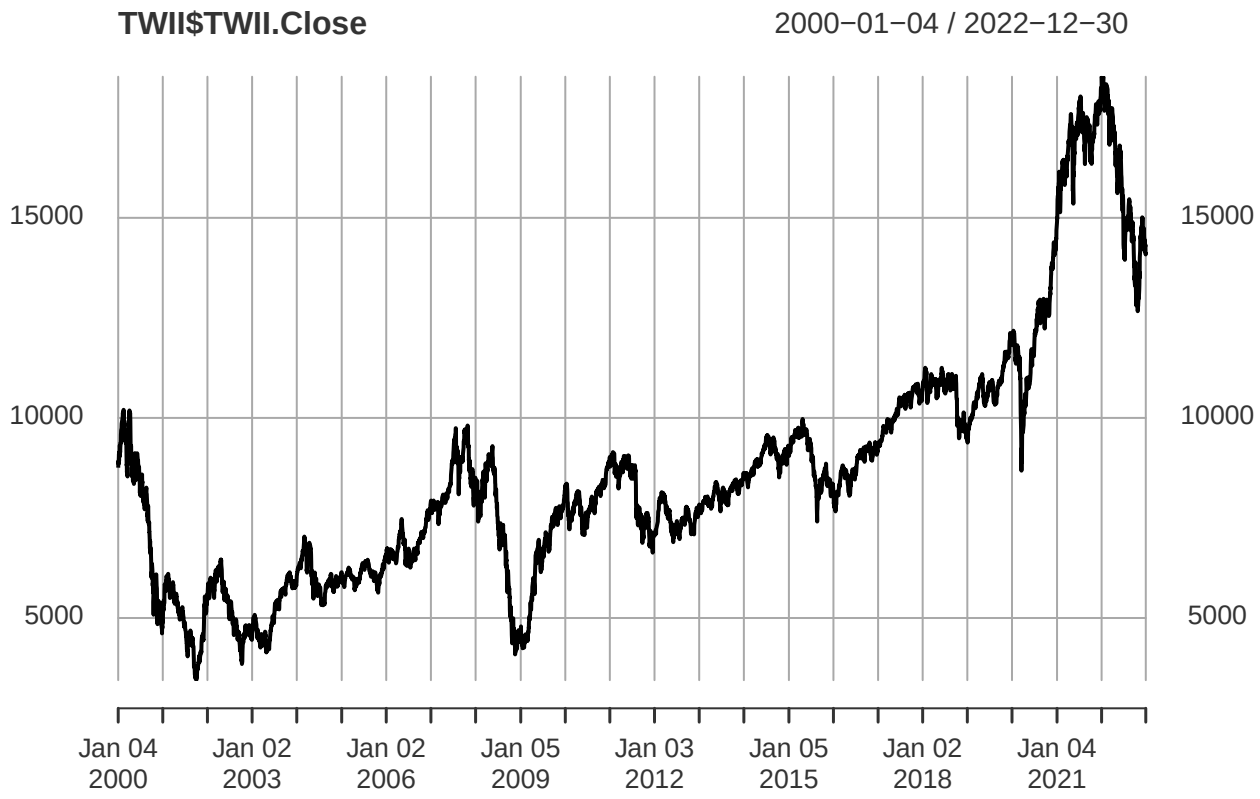
```
## Warning in adf.test(dlogGSPC): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dlogGSPC
## Dickey-Fuller = -18.219, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

### Plot of data




```
plot(TWII$TWII.Close)
```



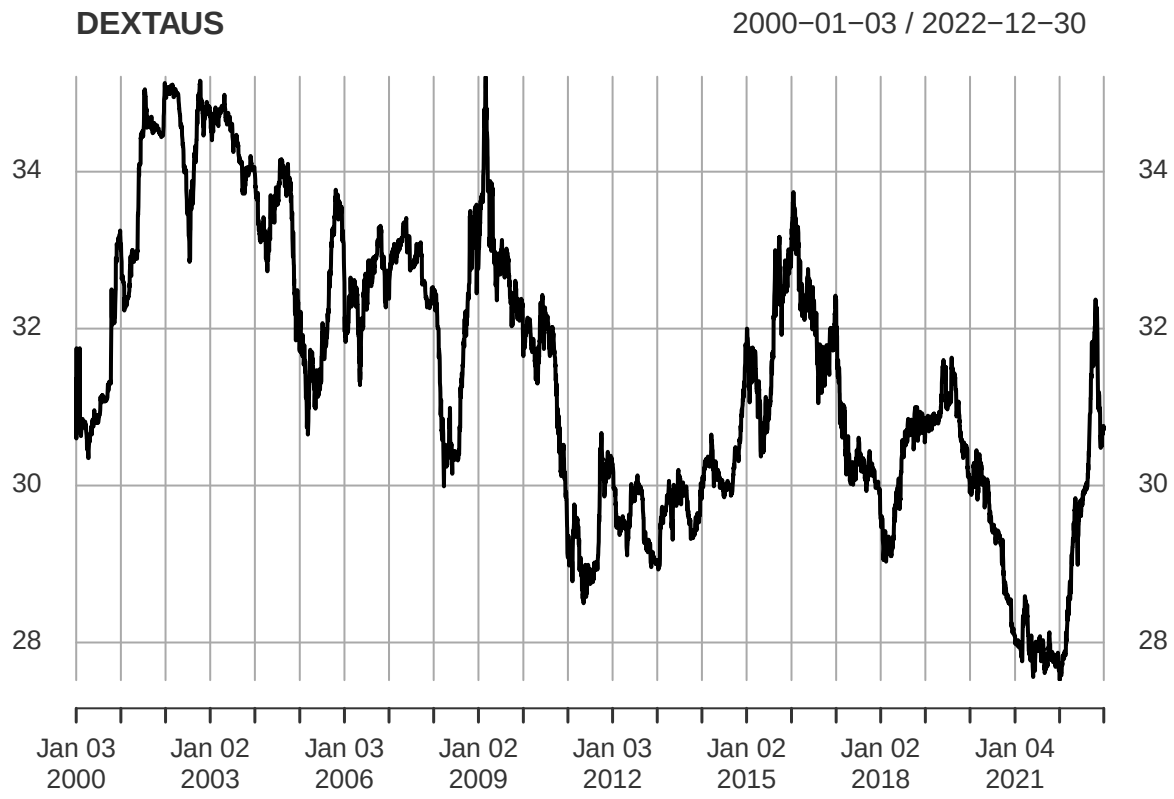

```
dev.copy(png, "Figure/TWII.png")
```

```
## quartz_off_screen
##                 3
```

```
dev.off()
```

```
## pdf
##   2
```

```
plot(DEXTAUS)
```



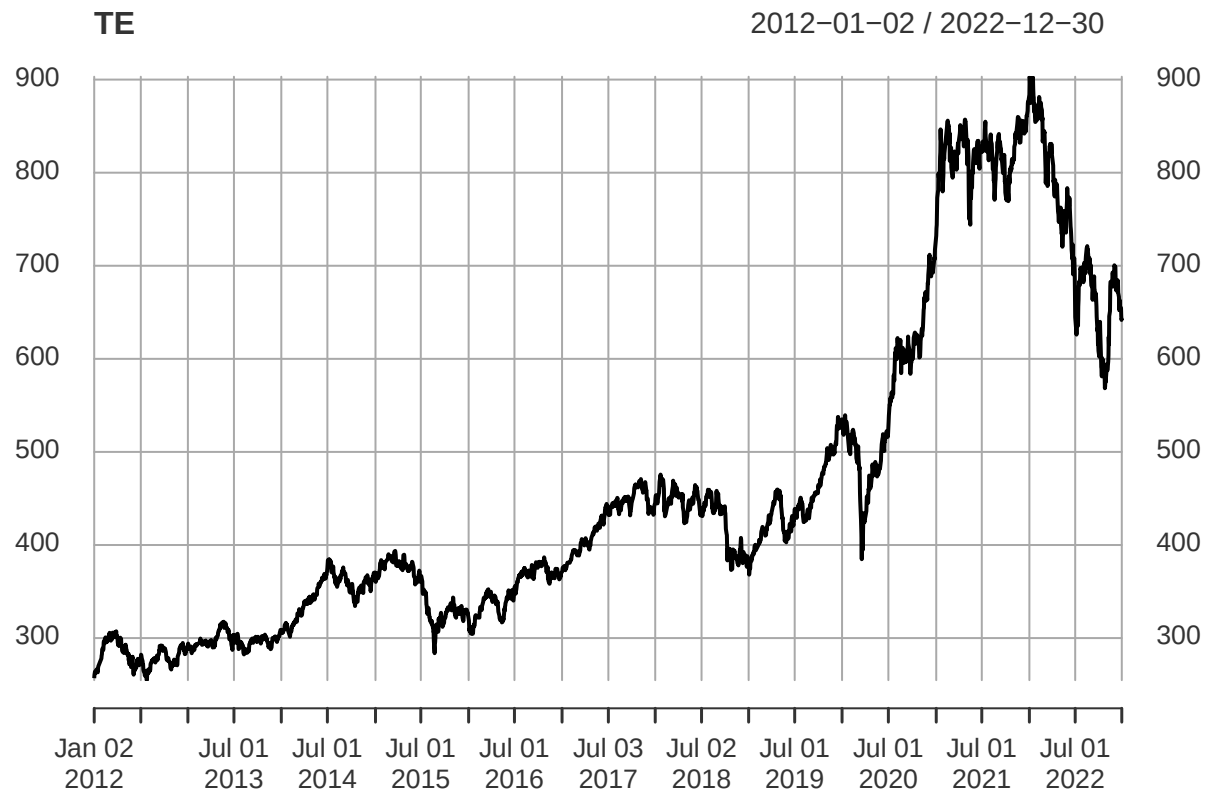

```
dev.copy(png, "Figure/DEXTAUS.png")
```

```
## quartz_off_screen
##                 3
```

```
dev.off()
```

```
## pdf
##   2
```

```
plot(TE)
```




```
dev.copy(png, "Figure/TE.png")
```

```
## quartz_off_screen
##                 3
```

```
dev.off()
```

```
## pdf
##   2
```

## Preparing for regression

Notice that our data have different lengths, which can cause problem when fitting regression.

When you use the `intersect()` function on two vectors of dates, the resulting object is a vector of date-times in `POSIXct` format. You can use the `as.Date()` or `as.POSIXct()` to convert to readable format.

However, here we have three time series to take intersection. If you have three `xts` objects and you want to find the intersection of their date ranges, you can use the `Reduce()` function in combination with the `intersect()` function.

```
date_dlogGSPC <- index(dlogGSPC)
date_dlogTWII <- index(dlogTWII)
date_dlogEX <- index(dlogEX)
date_dlogTE <- index(dlogTE)

common_date_TWII <- Reduce(intersect, list(date_dlogEX, date_dlogGSPC, date_dlogTWII)) |> as.Date()
common_date_TE <- Reduce(intersect, list(date_dlogEX, date_dlogGSPC, date_dlogTE)) |> as.Date()
```

```
dlogTWII_common <- dlogTWII[common_date_TWII]
dlogGSPC_common_TWII <- dlogGSPC[common_date_TWII]
x_TWII <- dlogTWII_common-dlogGSPC_common_TWII
y_TWII <- dlogEX[common_date_TWII]
```

```
dlogTE_common <- dlogTE[common_date_TE]
dlogGSPC_common_TE <- dlogGSPC[common_date_TE]
x_TE <- dlogTE_common-dlogGSPC_common_TE
y_TE <- dlogEX[common_date_TE]
```

For convenience, convert into data frame.

```
dlogTWII_common <- data.frame(dlogTWII_common)
dlogGSPC_common_TWII <- data.frame(dlogGSPC_common_TWII)
x_TWII <- data.frame(x_TWII)
y_TWII <- data.frame(y_TWII)
```

```
training_y_TWII <- head(y_TWII, -20)
training_x_TWII <- head(x_TWII,-20)

TWII_train_df <- data.frame(training_y_TWII, training_x_TWII)
```

```
dlogTE_common <- data.frame(dlogTE_common)
dlogGSPC_common_TE <- data.frame(dlogGSPC_common_TE)
x_TE <- data.frame(x_TE)
y_TE <- data.frame(y_TE)
```

```
training_y_TE <- head(y_TE, -20)
training_x_TE <- head(x_TE,-20)

TE_train_df <- data.frame(training_y_TE, training_x_TE)
```

## Preparing for Forecasts

```
testing_y_TWII <- tail(y_TWII, 20)
testing_x_TWII <- tail(x_TWII, 20)

TWII_test_df <- data.frame(testing_y_TWII, testing_x_TWII)
```

```
testing_y_TE <- tail(y_TE, 20)
testing_x_TE <- tail(x_TE, 20)

TE_test_df <- data.frame(testing_y_TE, testing_x_TE)
```

```
# For recovering
logEX_TWII <- logEX[common_date_TWII]
logEX_TE <- logEX[common_date_TE]

logEX_TWII <- data.frame(logEX_TWII)
logEX_TE <- data.frame(logEX_TE)
```

# Estimation

## In-sample regression

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v purrr     1.0.1
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lmtest)
library(sandwich)
```

```
N <- length(TWII_train_df$DEXTAUS)
m <- floor(0.75 * N^(1/3))
reg_TWII <- lm(DEXTAUS~ TWII.Close, data = TWII_train_df)
cat("OLS with Heteroskedasticity and Autocorrelation (HAC) Robust S.E.\n")
```

```
## OLS with Heteroskedasticity and Autocorrelation (HAC) Robust S.E.
```

```
hac_se_TWII <- coeftest(reg_TWII, vcov=NeweyWest(reg_TWII, prewhite = F, adjust = T, lag=m-1))
hac_se_TWII
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  2.8750e-06  4.0750e-05  0.0706    0.9438
## TWII.Close  -1.8806e-02  3.1656e-03 -5.9409 3.012e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
N <- length(TE_train_df$DEXTAUS)
m <- floor(0.75 * N^(1/3))
reg_TE <- lm(DEXTAUS ~ GSPC.Close, data = TE_train_df)
cat("OLS with Heteroskedasticity and Autocorrelation (HAC) Robust S.E.\n")
```

```
## OLS with Heteroskedasticity and Autocorrelation (HAC) Robust S.E.
```

```
hac_se_TE <- coeftest(reg_TE, vcov=NeweyWest(reg_TE, prewhite = F, adjust = T, lag=m-1))
hac_se_TE
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value Pr(>|t|)
## (Intercept) -4.0183e-06  5.3275e-05 -0.0754 0.939882
## GSPC.Close  -1.6841e-02  5.2700e-03 -3.1956 0.001413 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Random walk estimation

$y_t = y_{t-1} + w_t$ where $w_t \sim i.i.d.(0, \sigma^2)$

$\implies (1 - B)y_t = w_t$

```
lag_training_y_TWII <- lag(training_y_TWII)
# remove the first observation, since it has no lagged value
rw_training_y_TWII <- tail(training_y_TWII, -1)
lag_training_y_TWII <- tail(lag_training_y_TWII, -1)

rw_TWII_train_df <- data.frame(rw_training_y_TWII, lag=lag_training_y_TWII)

rw_TWII <- lm(DEXTAUS~ 0+DEXTAUS.1, data=rw_TWII_train_df)
rw_TWII
```

```
##
## Call:
## lm(formula = DEXTAUS ~ 0 + DEXTAUS.1, data = rw_TWII_train_df)
##
## Coefficients:
## DEXTAUS.1
##  -0.05814
```

```
lag_training_y_TE <- lag(training_y_TE)

# remove the first observation, since it has no lagged value
rw_training_y_TE <- tail(training_y_TE, -1)
lag_training_y_TE <- tail(lag_training_y_TE, -1)

rw_TE_train_df <- data.frame(rw_training_y_TE, lag=lag_training_y_TE)

rw_TE <- lm(DEXTAUS~ 0+DEXTAUS.1, data=rw_TE_train_df)
rw_TE
```

```
##
## Call:
## lm(formula = DEXTAUS ~ 0 + DEXTAUS.1, data = rw_TE_train_df)
##
## Coefficients:
## DEXTAUS.1
##  -0.03298
```

```
rw_TWII_arima <- arima(head(y_TWII, -20), order=c(0,1,0))
rw_TWII_arima
```

```
##
## Call:
## arima(x = head(y_TWII, -20), order = c(0, 1, 0))
##
##
## sigma^2 estimated as 1.955e-05:  log likelihood = 21619.96,  aic = -43237.92
```

```
rw_TE_arima <- arima(head(y_TE, -20), order=c(0,1,0))
rw_TE_arima
```

```
##
## Call:
## arima(x = head(y_TE, -20), order = c(0, 1, 0))
##
##
## sigma^2 estimated as 1.686e-05:  log likelihood = 10435.58,  aic = -20869.17
```

# Forecasting

```
library(forecast)
predict_TWII <- predict(reg_TWII, newdata = TWII_test_df)

# recover the predicted value
n <- 20
predict_TWII_EX <- predict_TWII + logEX_TWII[(nrow(logEX_TWII)-20+1):nrow(logEX_TWII)-1,]
```

```
predict_TE <- predict(reg_TE, newdata = TE_test_df)
# recover the predicted value
n <- 20
predict_TE_EX <- predict_TE + logEX_TE[(nrow(logEX_TE)-20+1):nrow(logEX_TE)-1,]
```

## MSE

```
mean((predict_TWII-tail(testing_y_TWII$DEXTAUS, 20))^2)
```

```
## [1] 5.305337e-06
```

```
mean((predict_TE-tail(testing_y_TE$DEXTAUS, 20))^2)
```

```
## [1] 5.221116e-06
```

## MSE from Random Walk

```
mean((testing_y_TWII$DEXTAUS)^2)
```

```
## [1] 5.357038e-06
```

```
mean((testing_y_TE$DEXTAUS)^2)
```

```
## [1] 5.357038e-06
```

# DM statistics

$d_t = (\hat{e}_{t+1}^{RW})^2 - (\hat{e}_{t+1}^{SRD})^2$

```
d_TWII <- (testing_y_TWII$DEXTAUS)^2-(predict_TWII-tail(testing_y_TWII$DEXTAUS, 20))^2

d_TE <-(testing_y_TE$DEXTAUS)^2- (predict_TE-tail(testing_y_TE$DEXTAUS, 20))^2
```

```
d_df <- data.frame(d_TWII, d_TE)
d_df
```

```
##                   d_TWII          d_TE
## 2022-12-02 -4.455908e-08 -5.883273e-09
## 2022-12-05  2.166419e-06  1.861828e-06
```

```
## 2022-12-06  5.104389e-07  9.347865e-07
## 2022-12-07 -1.323157e-07 -2.271976e-07
## 2022-12-08  5.754297e-07  5.122654e-07
## 2022-12-09  9.728758e-07  1.024521e-06
## 2022-12-12  1.371000e-06  1.308248e-06
## 2022-12-13 -7.311624e-07 -6.913064e-07
## 2022-12-14  3.161249e-06  3.320689e-06
## 2022-12-15 -3.868730e-06 -3.209537e-06
## 2022-12-16  3.651542e-07  6.305053e-07
## 2022-12-19 -3.029385e-08 -2.732031e-08
## 2022-12-20 -1.336338e-06 -1.365342e-06
## 2022-12-21 -4.200178e-07 -4.518547e-07
## 2022-12-22 -1.006649e-06 -8.778914e-07
## 2022-12-23  1.204698e-06  1.288956e-06
## 2022-12-27  1.527421e-07  1.421250e-07
## 2022-12-28 -6.406323e-08  9.446959e-08
## 2022-12-29 -1.648640e-06 -1.417022e-06
## 2022-12-30 -1.632147e-07 -1.266024e-07
```

```
N <- length(d_TWII)
m <- floor(0.75 * N^(1/3))
reg_d_TWII <- lm(d_TWII~1, data=d_df)
coeftest(reg_d_TWII, vcov=NeweyWest(reg_d_TWII, prewhite = F, adjust=T, lag=m-1))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.1701e-08 2.6651e-07   0.194   0.8482
```

```
N <- length(d_TE)
m <- floor(0.75 * N^(1/3))
reg_d_TE <- lm(d_TE~1, data=d_df)
coeftest(reg_d_TE, vcov=NeweyWest(reg_d_TE, prewhite = F, adjust=T, lag=m-1))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.3592e-07 2.4563e-07  0.5534   0.5865
```