IE423 Project Part II

Baran Kırkgöz & Orkan Çelikhisar

2023-12-11

Introduction

This assignment focuses on pairs trading by using control charts. For this purpose, time series data of BIST30 stock indices is used. To analyze correlations between two stock indices, two different approaches were applied: Constant Varience and Time Series Analysis. In the first task Constant varience assumption is used and in second task time series analysis approached is used.

First the libraries and packages are installed.

```
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("ggplot2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("tseries")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("quantmod")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("TTR")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("lubridate")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("forecast")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
install.packages("urca")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
install.packages("reshape2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## ############################ Warning from 'xts' package ##########################
## #                                                                                #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                             #
## #                                                                                #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                 #
## #                                                                                #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                                #
## ##################################################################################
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: TTR
```

```
library(TTR)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(forecast)
library(urca)
library(reshape2)
```

To analyze data, first they should be combined.

```
path <- "/cloud/project/data"
file_names <- list.files(path, pattern = "*.csv", full.names = TRUE)
all_data <- data.frame()
for (file in file_names) {
  temp_data <- read.csv(file, header = TRUE)
  all_data <- rbind(all_data, temp_data)
}
```

Then the timestamp is converted to date format and ensured that price is numeric, formatting shortname. Working in daily averages is healthier since the dataset is very large.

```
all_data$timestamp <- as.Date(all_data$timestamp)
all_data$price <- as.numeric(all_data$price)
all_data$short_name <- as.factor(all_data$short_name)

daily_avg_data <- all_data %>%
  group_by(timestamp, short_name) %>%
  summarize(daily_avg_price = mean(price))
```

```
## `summarise()` has grouped output by 'timestamp'. You can override using the
## `.groups` argument.
```

Splitting the data according to each stock:

```
stock_list <- split(daily_avg_data, daily_avg_data$short_name)
```

Preparing the data for correlation analysis: We need a matrix where each column is a time series of daily average prices for each stock First, create a list of data frames, each containing daily average prices for one stock

Adjusting stock_data_list to include timestamps and uniquely named price columns

```
stock_data_list <- lapply(names(stock_list), function(stock_name) {
  df <- stock_list[[stock_name]]
  df <- df[order(df$timestamp), ]  # Ensure data is ordered by date
  colnames(df)[colnames(df) == "daily_avg_price"] <- paste0(stock_name, "_price")  # Rename price colum
  return(df[, c("timestamp", paste0(stock_name, "_price"))])  # Keep only timestamp and renamed price c
})
```

Aligning the time series: Now each data frame in the list has a 'timestamp' and one uniquely named price column

```
aligned_stock_data <- Reduce(function(x, y) merge(x, y, by = "timestamp", all = TRUE), stock_data_list)
```

We stop for a moment to take a look at how beautiful aligned_stock_data is. We adore our creation.

Removing rows with NA.

```
aligned_stock_data <- na.omit(aligned_stock_data)
```

Convert the aligned data to matrix for correlation analysis, ignoring the 'timestamp' column.

```
stock_price_matrix <- as.matrix(aligned_stock_data[, -1])
```

Compute pairwise correlations:

```
cor_matrix <- cor(stock_price_matrix, use = "pairwise.complete.obs")
```

Now we're going to find the two highest correlated pairs:

Extracting the stock names from the columns of the stock_price_matrix

```
stock_names <- colnames(stock_price_matrix)
```

Identify all unique pairs and their correlation coefficients:

```
pair_correlations <- data.frame(pair1 = numeric(), pair2 = numeric(), correlation = numeric())
```

Loop through the correlation matrix and record each pair:

```
for (i in 1:(nrow(cor_matrix) - 1)) {
  for (j in (i + 1):ncol(cor_matrix)) {
    pair_correlations <- rbind(pair_correlations,
                               data.frame(pair1 = stock_names[i],
                                          pair2 = stock_names[j],
                                          correlation = cor_matrix[i, j]))
  }
}
```

Sorting the pairs by correlation in descending order:

```
sorted_pairs <- pair_correlations[order(-pair_correlations$correlation), ]
```

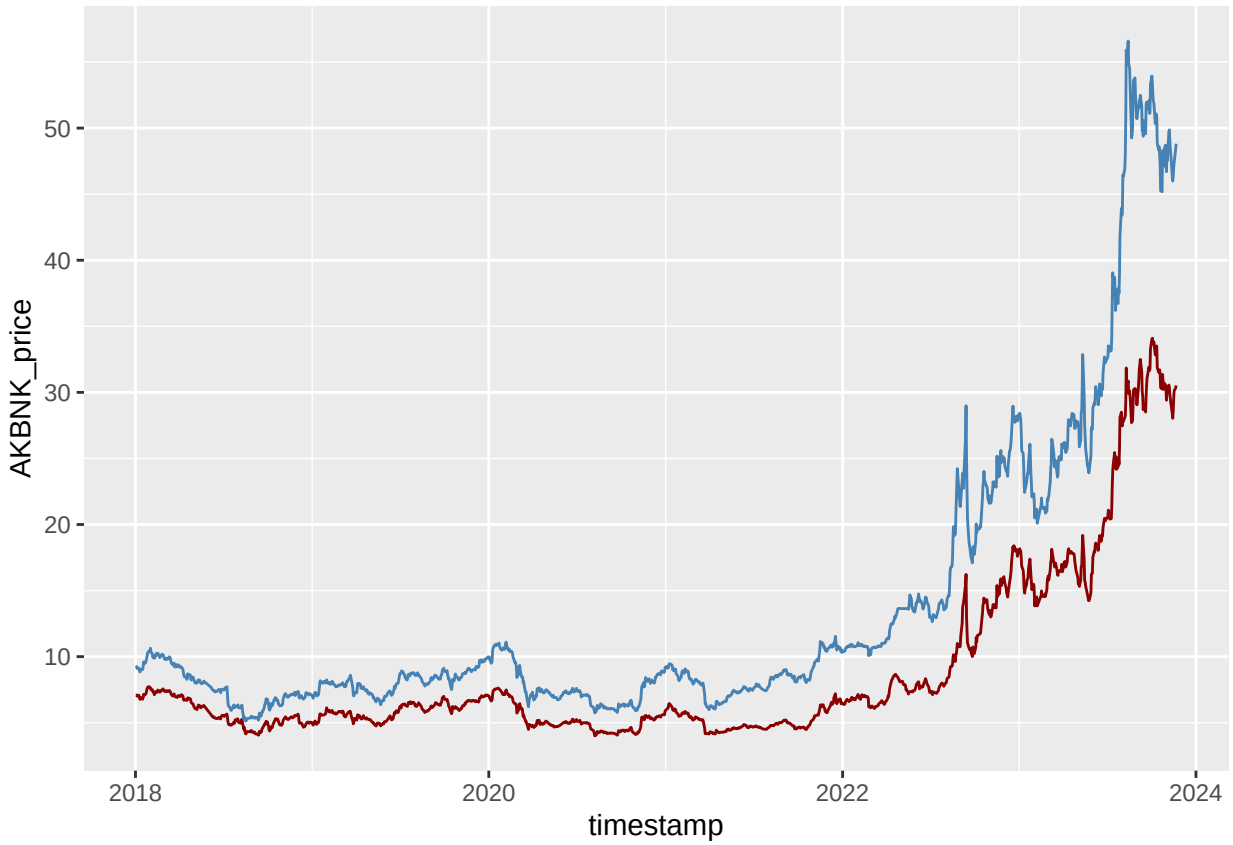
Selecting the top two pairs:

```
top_two_pairs <- head(sorted_pairs, 2)
print(top_two_pairs)
```

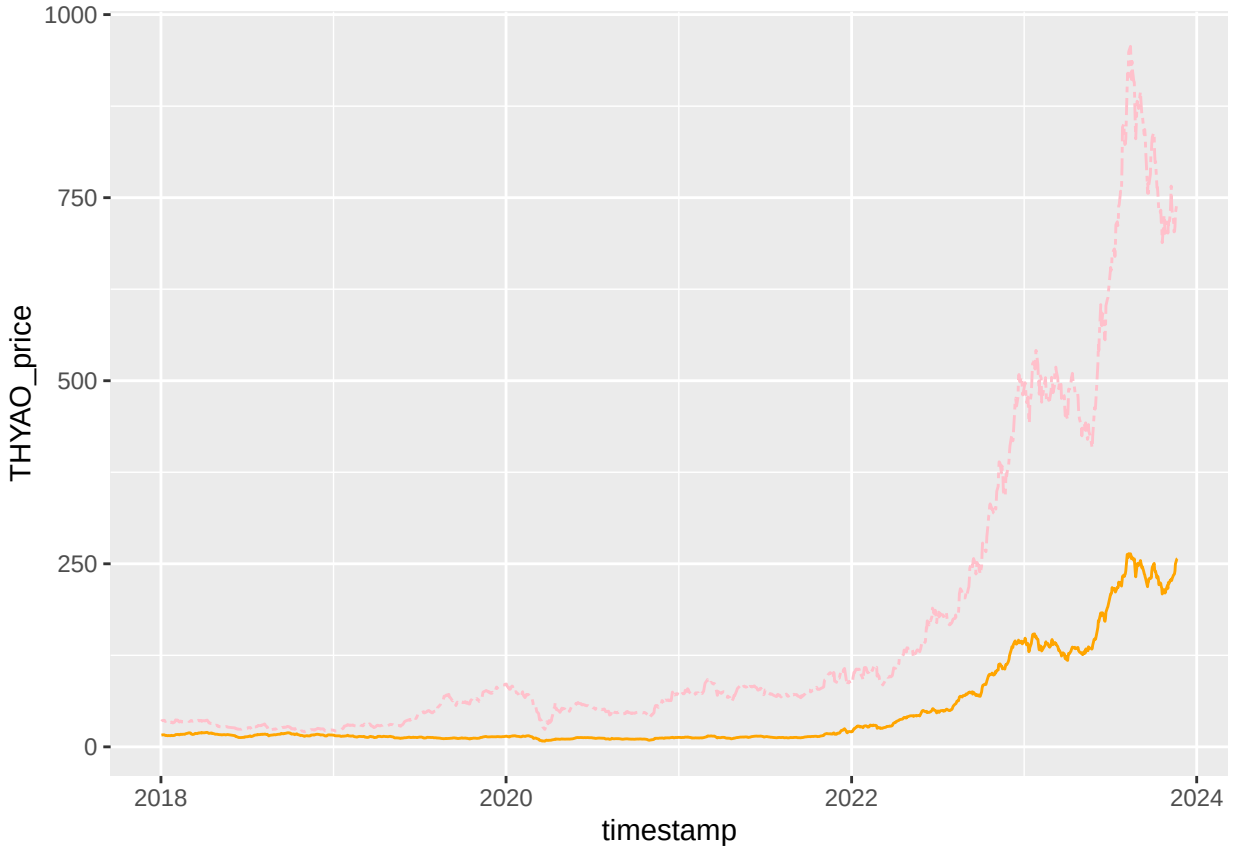
```
##           pair1       pair2 correlation
## 363 PGSUS_price THYAO_price   0.9931033
## 8   AKBNK_price GARAN_price   0.9927360
```

Our top two correlated pairs are Akbank and Garanti & THY and Pegasus. Both of them have correlation relationship larger than 0.99. This makes a lot of sense considering they are very similar companies operating in the same industries. To understand the correlation better, the graphs are plotted.

```
ggplot(aligned_stock_data, aes(x=timestamp)) +
  geom_line(aes(y = AKBNK_price), color = "darkred") +
  geom_line(aes(y = GARAN_price), color="steelblue", linetype="solid")
```




```
ggplot(aligned_stock_data, aes(x=timestamp)) +
  geom_line(aes(y = THYAO_price), color = "orange") +
  geom_line(aes(y = PGSUS_price), color="pink", linetype="twodash")
```



From both graph, the strong correlation can easily be seen, the lines are moving together.

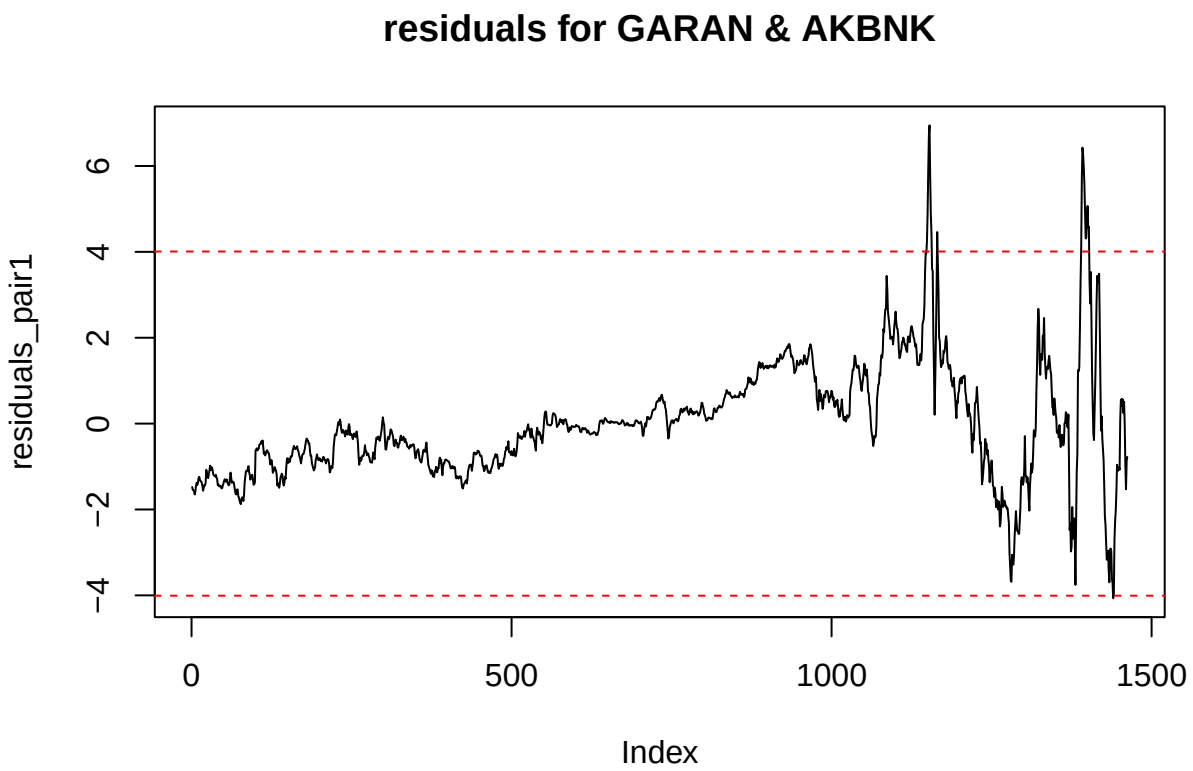
Now, we will be applying linear regression modeling to these stock pairs Extract the relevant columns for the first pair

```
pair1_data <- aligned_stock_data[, c("AKBNK_price", "GARAN_price")]
pair2_data <- aligned_stock_data[, c("PGSUS_price", "THYAO_price")]
```

Calculating control limits for residuals (assuming constant variance) Example: using 3 standard deviations (natural limits) as control limits

Linear Regression for Pair 1 (AKBNK and GARAN)

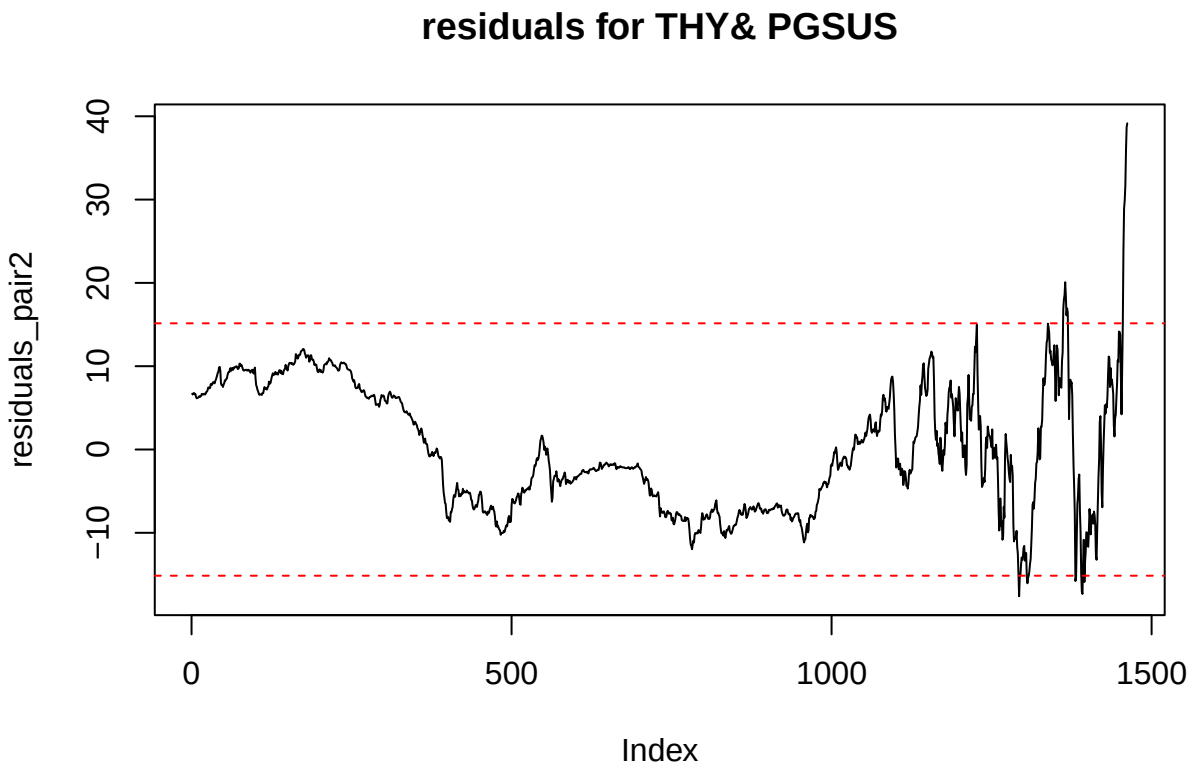
```
model_pair1 <- lm(GARAN_price ~ AKBNK_price, data = pair1_data)
residuals_pair1 <- resid(model_pair1)
control_limit_pair1 <- 3 * sd(residuals_pair1)
plot(residuals_pair1, type= "l", main="residuals for GARAN & AKBNK")
abline(h = control_limit_pair1, col = "red", lty = 2)
abline(h = (-1)*control_limit_pair1, col = "red", lty = 2)
```



From the control chart the points located below lower control limit and above upper control limit can be detected. This points represents potential trading oppurtunity.

Linear Regression for Pair 2 (PGSUS and THYAO)

```
model_pair2 <- lm(THYAO_price ~ PGSUS_price, data = pair2_data)
residuals_pair2 <- resid(model_pair2)

control_limit_pair2 <- 2 * sd(residuals_pair2) #since the limits are too big and 3 sigma limits don't t
plot(residuals_pair2, type= "l", main="residuals for THY& PGSUS")
abline(h = control_limit_pair2, col = "red", lty = 2)
abline(h = (-1)*control_limit_pair2, col = "red", lty = 2)
```



Similarly, from the control chart we detect some points below&above control limits which represents an alarm for strong trading oppurtunity.

— Trading Algorithm Logic—

We initialize variables to store entry prices for both long and short positions. The capital is split evenly between the two stocks in the pair. When a trade is entered, the number of shares for each stock is calculated based on half the capital and the current stock price. A short position is represented by a negative number of shares, while a long position is positive. When exiting the trade, the profit is calculated based on the change in stock prices multiplied by the number of shares held. The positions are reset to zero after closing the trade. We open positions when the residual crosses the control limit and close them if it returns within the control. The algorithm closes the position in 'h' consecutive hits in the control limits. We can adjust 'h' below. 3 hits proved to be most profitable. The final capital and total profit for Pair 1 are calculated and printed.

Trading Simulation for Pair 1 (GARAN_price ~ AKBNK_price)

```
capital_pair1 <- 10000  # Starting capital for Pair 1
profit_pair1 <- 0  # Profit from trading Pair 1
```

Initialize variables to store entry prices, positions, and count consecutive control limit hits

```
position_GARAN <- 0
position_AKBNK <- 0
control_limit_hit_count <- 0  # Counter for consecutive control limit hits
h <- 3 # Indicate here after  how many hits in the control limits would you like to close the position

for (i in 2:length(residuals_pair1)) {
  # Check if the residual returns within the control limits
  if (residuals_pair1[i] <= control_limit_pair1 && residuals_pair1[i] >= -control_limit_pair1) {
    control_limit_hit_count <- control_limit_hit_count + 1
```

```
  } else {
    control_limit_hit_count <- 0  # Reset count if out of control limits
  }

  # Check for entry conditions
  if (residuals_pair1[i-1] <= control_limit_pair1 && residuals_pair1[i] > control_limit_pair1 && positi
    # Entry point for trade: GARAN is overpriced relative to AKBNK
    position_GARAN <- -capital_pair1 / 2 / pair1_data$GARAN_price[i]  # Short position for GARAN
    position_AKBNK <- capital_pair1 / 2 / pair1_data$AKBNK_price[i]   # Long position for AKBNK
  } else if (residuals_pair1[i-1] >= -control_limit_pair1 && residuals_pair1[i] < -control_limit_pair1 &
    # Entry point for trade: GARAN is underpriced relative to AKBNK
    position_GARAN <- capital_pair1 / 2 / pair1_data$GARAN_price[i]   # Long position for GARAN
    position_AKBNK <- -capital_pair1 / 2 / pair1_data$AKBNK_price[i]  # Short position for AKBNK
  }

  # Check for exit conditions (10 consecutive times within control limits)
  if (control_limit_hit_count >= h) {
    if (position_GARAN != 0 || position_AKBNK != 0) {
      # Close positions and calculate profit
      close_price_GARAN <- pair1_data$GARAN_price[i]
      close_price_AKBNK <- pair1_data$AKBNK_price[i]
      profit_pair1 <- profit_pair1 + position_GARAN * (close_price_GARAN - pair1_data$GARAN_price[i-1])
      position_GARAN <- 0
      position_AKBNK <- 0
    }
    control_limit_hit_count <- 0  # Reset count after closing positions
  }
}
# Update the final capital with the profit from Pair 1
final_capital_pair1 <- capital_pair1 + profit_pair1
```

Print results for Pair 1

```
print(paste("Final Capital for Pair 1:", final_capital_pair1))
```

```
## [1] "Final Capital for Pair 1: 10667.9488938846"
```

```
print(paste("Total Profit for Pair 1:", profit_pair1))
```

```
## [1] "Total Profit for Pair 1: 667.948893884581"
```

BAM! We just made 667 Turkish Liras in 4 years and didn't even do any labor.

Trading Simulation for Pair 2 (PGSUS_price ~ THYAO_price)

```
capital_pair2 <- 10000  # Starting capital for Pair 2
profit_pair2 <- 0  # Profit from trading Pair 2
```

Initialize variables to store positions and count consecutive control limit hits

```
position_PGSUS <- 0
position_THYAO <- 0
control_limit_hit_count <- 0  # Counter for consecutive control limit hits
h <- 1 # Indicate here after how many hits in the control limits would you like to close the position
```

```
for (i in 2:length(residuals_pair2)) {
  # Check if the residual returns within the control limits
  if (residuals_pair2[i] <= control_limit_pair2 && residuals_pair2[i] >= -control_limit_pair2) {
```

```
    control_limit_hit_count <- control_limit_hit_count + 1
  } else {
    control_limit_hit_count <- 0  # Reset count if out of control limits
  }

  # Check for entry conditions
  if (residuals_pair2[i-1] <= control_limit_pair2 && residuals_pair2[i] > control_limit_pair2 && positi
    # Entry point for trade: PGSUS is overpriced relative to THYAO
    position_PGSUS <- -capital_pair2 / 2 / pair2_data$PGSUS_price[i]  # Short position for PGSUS
    position_THYAO <- capital_pair2 / 2 / pair2_data$THYAO_price[i]   # Long position for THYAO
  } else if (residuals_pair2[i-1] >= -control_limit_pair2 && residuals_pair2[i] < -control_limit_pair2 &
    # Entry point for trade: PGSUS is underpriced relative to THYAO
    position_PGSUS <- capital_pair2 / 2 / pair2_data$PGSUS_price[i]   # Long position for PGSUS
    position_THYAO <- -capital_pair2 / 2 / pair2_data$THYAO_price[i]  # Short position for THYAO
  }

  # Check for exit conditions (h consecutive times within control limits)
  if (control_limit_hit_count >= h) {
    if (position_PGSUS != 0 || position_THYAO != 0) {
      # Close positions and calculate profit
      close_price_PGSUS <- pair2_data$PGSUS_price[i]
      close_price_THYAO <- pair2_data$THYAO_price[i]
      profit_pair2 <- profit_pair2 + position_PGSUS * (close_price_PGSUS - pair2_data$PGSUS_price[i-1])
      position_PGSUS <- 0
      position_THYAO <- 0
    }
    control_limit_hit_count <- 0  # Reset count after closing positions
  }
}

# Update the final capital with the profit from Pair 2
final_capital_pair2 <- capital_pair2 + profit_pair2
```

Print results for Pair 2

```
print(paste("Final Capital for Pair 2:", final_capital_pair2))
```

```
## [1] "Final Capital for Pair 2: 9615.65425177567"
```

```
print(paste("Total Profit for Pair 2:", profit_pair2))
```

```
## [1] "Total Profit for Pair 2: -384.345748224327"
```

We have unfortunately lost 384 Turkish Liras in 4 years. Wall Street won this time.

Task 2: Advanced Pairs Trading Strategy Using Time Series Analysis

For the advanced pairs trading strategy using time series analysis, we will incorporate methods such as Autoregressive Integrated Moving Average (ARIMA), Cointegration, or similar time series methodologies to better understand and model the residuals of the selected stock pairs. This advanced approach aims to capture more complex dynamics in the residuals than the basic strategy, potentially leading to improved trading decisions.

Time Series Analysis for Pair 1 & 2

```
ts_resid_pair1 <- ts(residuals_pair1)
ts_resid_pair2 <- ts(residuals_pair2)
```

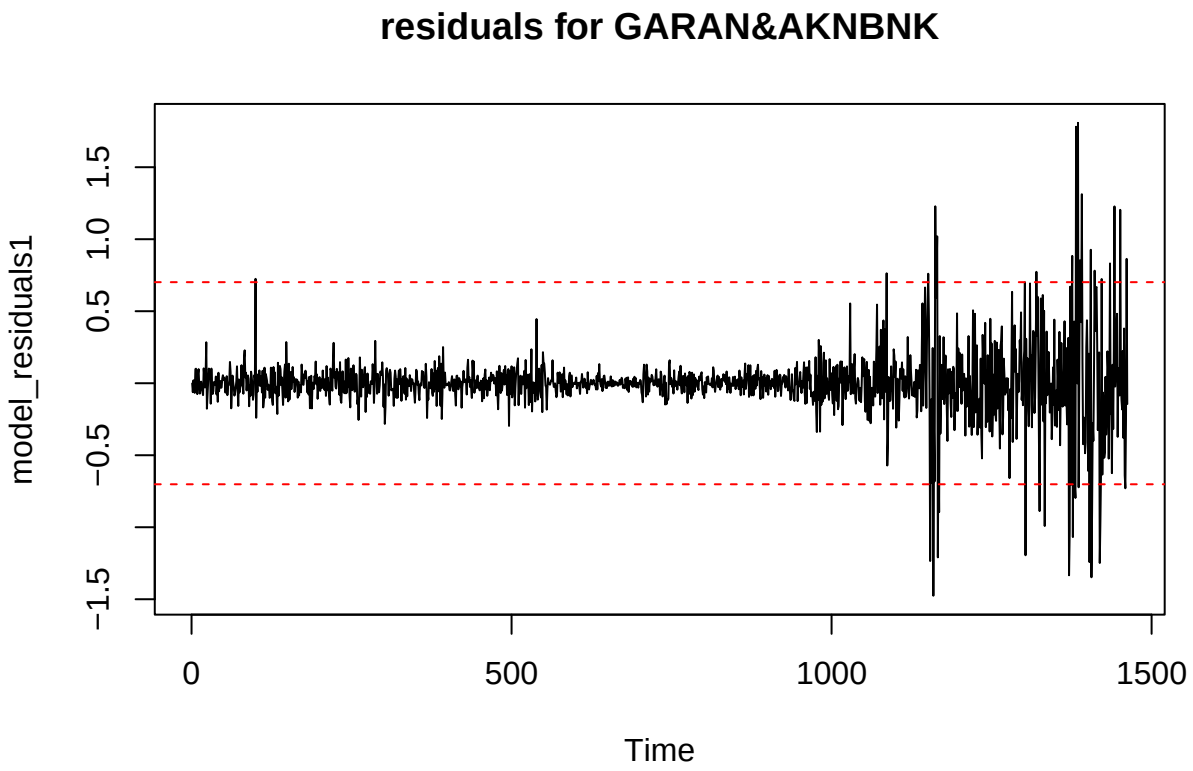
Fitting ARIMA or other suitable model

```
model_ts_pair1 <- auto.arima(ts_resid_pair1)
model_ts_pair2 <- auto.arima(ts_resid_pair2)
```

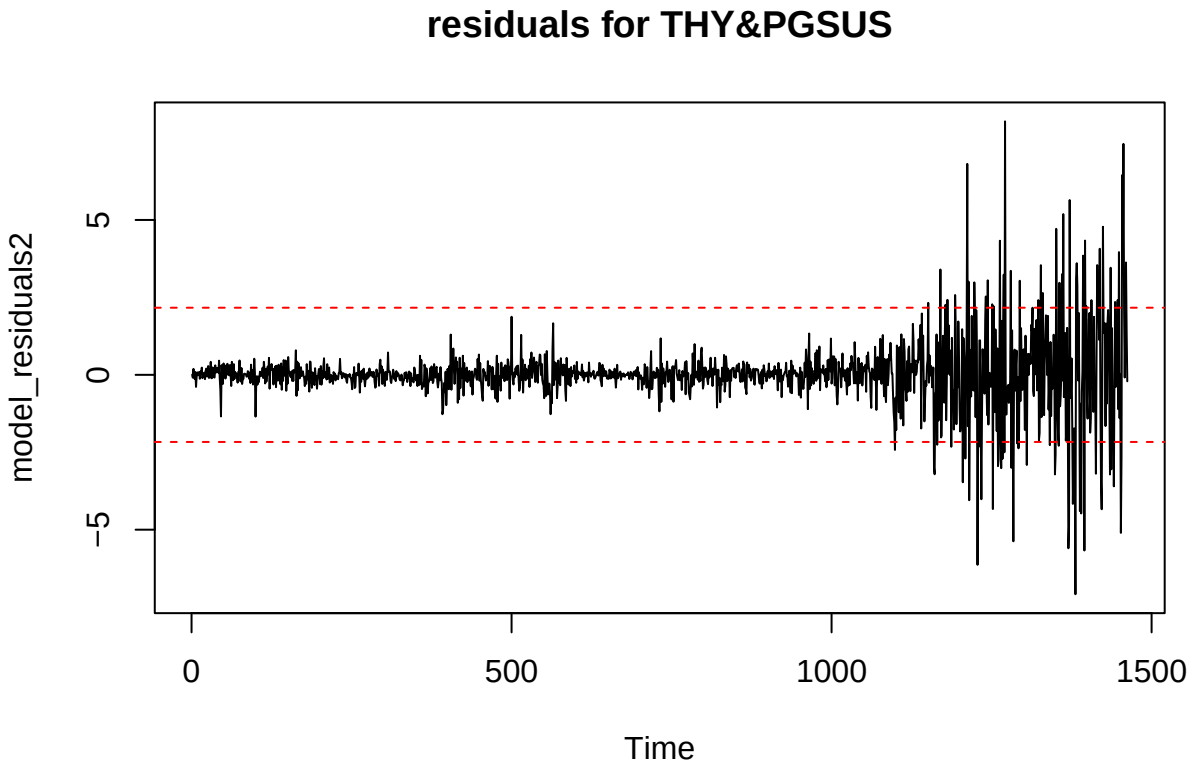
Redefining control limits based on the new model

```
model_residuals1 <- residuals(model_ts_pair1)
model_residuals2 <- residuals(model_ts_pair2)

std_dev_residuals1 <- sd(model_residuals1)
std_dev_residuals2 <- sd(model_residuals2)

control_limit_pair12 <- 3 * std_dev_residuals1
control_limit_pair22 <- 2 * std_dev_residuals2
```

```
plot(model_residuals1, type= "l", main="residuals for GARAN&AKNBNK")
abline(h = control_limit_pair12, col = "red", lty = 2)
abline(h = (-1)*control_limit_pair12, col = "red", lty = 2)
```




```
plot(model_residuals2, type= "l", main="residuals for THY&PGSUS")
abline(h = control_limit_pair22, col = "red", lty = 2)
abline(h = (-1)*control_limit_pair22, col = "red", lty = 2)
```



In this second model, the control limits for both of the charts become narrower. This is mainly because of the constant varience assumption. Since ARIMA model works with moving average and autoregression methods, the varience of the residuals are much lower than the simple linear regression. This makes ARIMA model stronger in detecting trade opportunities. ## —- Running the Trading Simulation Again —- Trading Simulation for Pair 1 (GARAN_price ~ AKBNK_price)

```
capital_pair1 <- 10000  # Starting capital for Pair 1
profit_pair1 <- 0  # Profit from trading Pair 1
```

Initialize variables to store entry prices, positions, and count consecutive control limit hits

```
position_GARAN <- 0
position_AKBNK <- 0
control_limit_hit_count <- 0  # Counter for consecutive control limit hits
h <- 3 # Indicate here after  how many hits in the control limits would you like to close the position

for (i in 2:length(residuals_pair1)) {
  # Check if the residual returns within the control limits
  if (residuals_pair1[i] <= control_limit_pair12 && residuals_pair1[i] >= -control_limit_pair12) {
    control_limit_hit_count <- control_limit_hit_count + 1
  } else {
    control_limit_hit_count <- 0  # Reset count if out of control limits
  }

  # Check for entry conditions
  if (residuals_pair1[i-1] <= control_limit_pair12 && residuals_pair1[i] > control_limit_pair12 && posi
    # Entry point for trade: GARAN is overpriced relative to AKBNK
    position_GARAN <- -capital_pair1 / 2 / pair1_data$GARAN_price[i]  # Short position for GARAN
    position_AKBNK <- capital_pair1 / 2 / pair1_data$AKBNK_price[i]   # Long position for AKBNK
  } else if (residuals_pair1[i-1] >= -control_limit_pair12 && residuals_pair1[i] < -control_limit_pair1
    # Entry point for trade: GARAN is underpriced relative to AKBNK
```

```
    position_GARAN <- capital_pair1 / 2 / pair1_data$GARAN_price[i]   # Long position for GARAN
    position_AKBNK <- -capital_pair1 / 2 / pair1_data$AKBNK_price[i]  # Short position for AKBNK
  }

  # Check for exit conditions (10 consecutive times within control limits)
  if (control_limit_hit_count >= h) {
    if (position_GARAN != 0 || position_AKBNK != 0) {
      # Close positions and calculate profit
      close_price_GARAN <- pair1_data$GARAN_price[i]
      close_price_AKBNK <- pair1_data$AKBNK_price[i]
      profit_pair1 <- profit_pair1 + position_GARAN * (close_price_GARAN - pair1_data$GARAN_price[i-1])
      position_GARAN <- 0
      position_AKBNK <- 0
    }
    control_limit_hit_count <- 0  # Reset count after closing positions
  }
}
```

Update the final capital with the profit from Pair 1

```
final_capital_pair1 <- capital_pair1 + profit_pair1
```

Print results for Pair 1

```
print(paste("Final Capital for Pair 1:", final_capital_pair1))
```

```
## [1] "Final Capital for Pair 1: 10600.5281057596"
```

```
print(paste("Total Profit for Pair 1:", profit_pair1))
```

```
## [1] "Total Profit for Pair 1: 600.528105759592"
```

Trading Simulation for Pair 2 (PGSUS_price ~ THYAO_price)

```
capital_pair2 <- 10000  # Starting capital for Pair 2
profit_pair2 <- 0  # Profit from trading Pair 2
```

Initialize variables to store positions and count consecutive control limit hits

```
position_PGSUS <- 0
position_THYAO <- 0
control_limit_hit_count <- 0  # Counter for consecutive control limit hits
h <- 1 # Indicate here after how many hits in the control limits would you like to close the position

for (i in 2:length(residuals_pair2)) {
  # Check if the residual returns within the control limits
  if (residuals_pair2[i] <= control_limit_pair22 && residuals_pair2[i] >= -control_limit_pair22) {
    control_limit_hit_count <- control_limit_hit_count + 1
  } else {
    control_limit_hit_count <- 0  # Reset count if out of control limits
  }

  # Check for entry conditions
  if (residuals_pair2[i-1] <= control_limit_pair22 && residuals_pair2[i] > control_limit_pair22 && posi
    # Entry point for trade: PGSUS is overpriced relative to THYAO
    position_PGSUS <- -capital_pair2 / 2 / pair2_data$PGSUS_price[i]  # Short position for PGSUS
    position_THYAO <- capital_pair2 / 2 / pair2_data$THYAO_price[i]   # Long position for THYAO
  } else if (residuals_pair2[i-1] >= -control_limit_pair22 && residuals_pair2[i] < -control_limit_pair2
```

```
    # Entry point for trade: PGSUS is underpriced relative to THYAO
    position_PGSUS <- capital_pair2 / 2 / pair2_data$PGSUS_price[i]   # Long position for PGSUS
    position_THYAO <- -capital_pair2 / 2 / pair2_data$THYAO_price[i]  # Short position for THYAO
  }

  # Check for exit conditions (h consecutive times within control limits)
  if (control_limit_hit_count >= h) {
    if (position_PGSUS != 0 || position_THYAO != 0) {
      # Close positions and calculate profit
      close_price_PGSUS <- pair2_data$PGSUS_price[i]
      close_price_THYAO <- pair2_data$THYAO_price[i]
      profit_pair2 <- profit_pair2 + position_PGSUS * (close_price_PGSUS - pair2_data$PGSUS_price[i-1])
      position_PGSUS <- 0
      position_THYAO <- 0
    }
    control_limit_hit_count <- 0  # Reset count after closing positions
  }
}
```

Update the final capital with the profit from Pair 2

```
final_capital_pair2 <- capital_pair2 + profit_pair2
```

Print results for Pair 2

```
print(paste("Final Capital for Pair 2:", final_capital_pair2))
```

```
## [1] "Final Capital for Pair 2: 5815.06512975546"
```

```
print(paste("Total Profit for Pair 2:", profit_pair2))
```

```
## [1] "Total Profit for Pair 2: -4184.93487024454"
```

Conclusion

Although second task includes more sensitive method, we generated more profit with the simple linear regression model. The reason behind is that we capture more out of control points with ARIMA model and it decreases the profit. But the profit that we generate with simple linear regression model is not large enough, which shows checking only correlations between pairs is not effective enough.