Graphical LASSO for Pairs Trading

Federico Russo, Bocconi University
Changchen Yu, Bocconi University

December 2022

Abstract

In this paper we investigate the use of Graphical LASSO (GLASSO) for pairs trading, a trading strategy based on statistical arbitrage that aims to explore mean-reversion patterns between pairs of financial securities. We propose a model selection method that, to our knowledge, has never been adopted in this context. The **research question** is whether a GLASSO-based pairs trading strategy can outperform strategies that select pairs randomly.

1 Data

As for the stock quotes, we perform Web Scraping from *Yahoo Finance*, obtaining daily data for the 40 components of the Italian stock index FTSE MIB, from 2014 to 2022. Moreover, we match each stock to its sector by scraping the official website of *Borsa Italiana* (Italian Stock Exchange). The main assumption of GLASSO is that the input data follows a multivariate normal distribution. Nevertheless, neither stock prices nor their returns are known to follow such distribution, but log returns do. We define log return with period τ as:

$$r(t) = ln(\frac{S_t}{S_{t-\tau}}) = ln(S_t) - ln(S_{t-\tau}),$$

where S_t is the stock price at day t. We therefore apply this transformation to our data and perform a Shapiro-Wilk normality test with $\alpha = 0.01$, as proposed by Mota (2012), obtaining the following:

Year	Normality % (daily)	Normality % (weekly)
2014	48,48	87,88
2015	48,57	97,14
2016	8,11	83,78
2017	21,05	86,84
2018	23,68	92,11
2019	25,64	94,87
2020	0,00	43,59
2021	15,38	79,49

Figure 1: Percentage of stocks that do not reject the null hypothesis (normality).

On average, the vast majority of the stocks do not reject normality hypothesis when $\tau = 7$ is adopted. The unsatisfactory results for daily returns might be due to the fact that the sample size ($N = 365$) is too big compared to the optimal one for Shapiro-Wilk test, which according to Ahad (2011) is between $N = 3$ and $N = 50$. In 2020, due to high market volatility and frequent tail events, the normal distribution did not manage to fit the data, independently of the choice of τ. For this reason, we exclude this year as well as 2021 from our analysis. After performing this exclusion, we consider the normality assumption to be satisfied when $\tau = 7$.

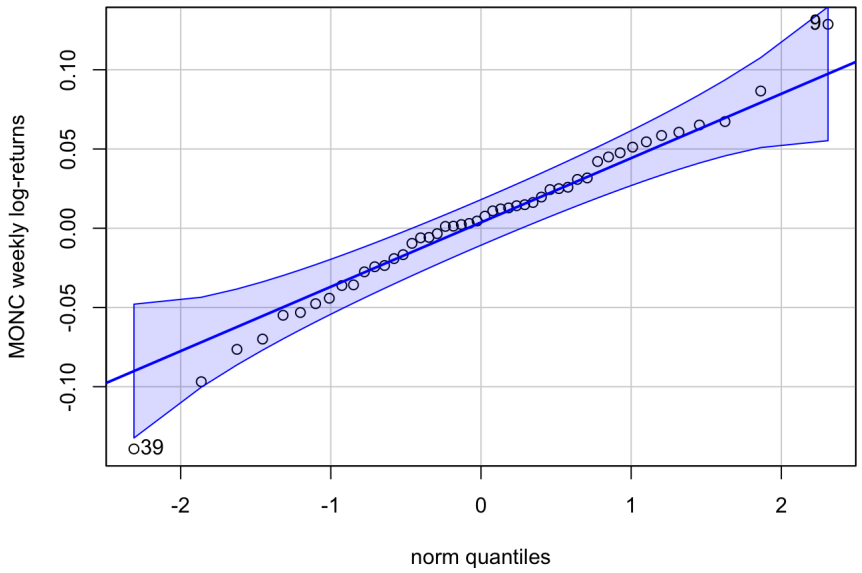

Figure 2: QQPlot of weekly log-returns of MONC stock in 2015.

2 Graphical LASSO

In many financial problems it is crucial to understand the relations between different assets, which are usually measured with correlation. Nevertheless, if two assets belong to the same index (as in the case of two stocks of the FTSE MIB), they might possess a strong spurious correlation which is not explained by any economic rationale. This is due to their common correlation with the index they belong to: the so-called *market beta*, that acts as a confounding variable. GLASSO by calculating the precision matrix, exploits the partial correlation between the stocks, thus isolating the market beta effect and discovering hidden relationships.

As proposed by Robot Wealth (2020), we standardise weekly log returns before calculating their covariance matrix over the period 2014-2016. This matrix becomes one of the two inputs of the GLASSO (*glasso* package in R), together with the parameter $\rho \in [0, 1]$, that regulates the sparsity of the precision matrix. Next, we calculate the partial correlation matrix PC according to the formula:

$$\forall i \neq j \; PC_{i,j} = \frac{-\Omega_{i,j}}{\sqrt{\Omega_{i,i} \cdot \Omega_{j,j}}}, \; PC_{i,i} = 0,$$

where Ω represents the precision matrix calculated by the GLASSO. Finally, the partial correlations can be represented as an undirected graph where each stock is a node and each non-zero correlation between two stocks is an edge.

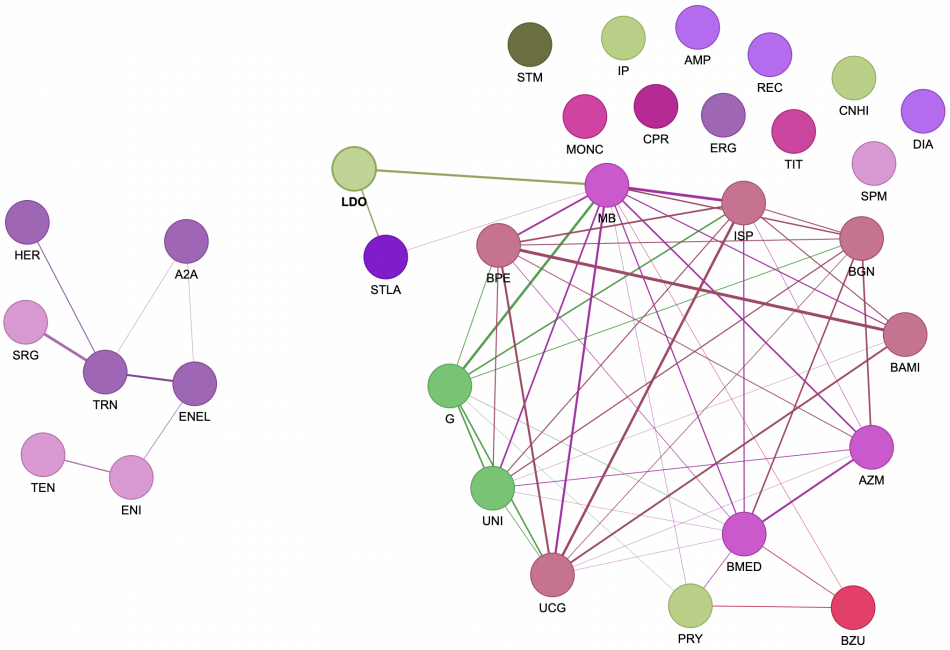

Figure 3: Graph of the partial correlations ($\rho = 0.63$). On the left there are energy and utilities companies, while the cluster of financial institutions and insurance companies is on the right there.

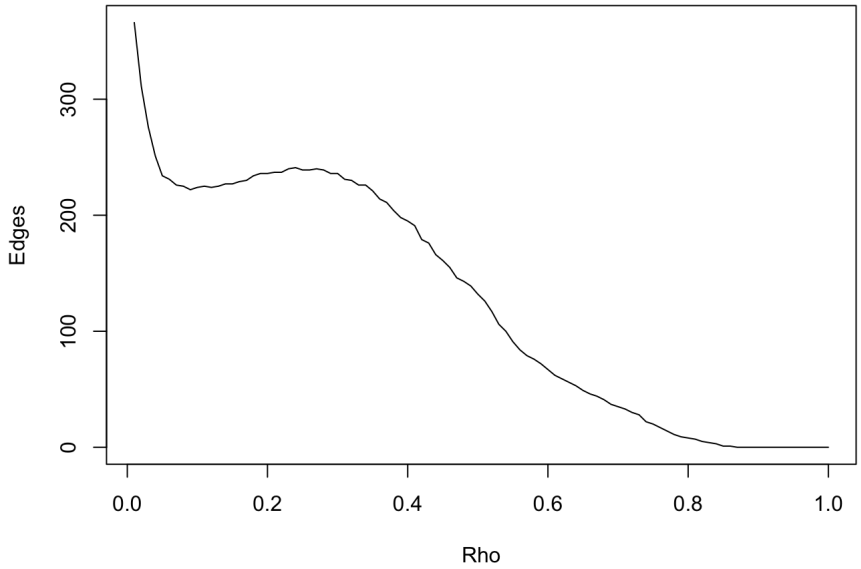

Figure 4: Number of edges in the graph corresponding to different ρ values in the period 2014-2016.

3 An Application to Pairs Trading

In a pairs trading strategy, two securities X and Y are traded whenever the Z-score of the spread $Y - \beta X$, where β is calculated by performing a rolling linear regression of Y on X, diverges from its long-term mean and crosses a certain threshold for which a trading signal is emitted. Since coding a pairs trading strategy is outside of the scope of this paper, we rely on the *PairTrading* package in R to do so. In this package, the strategy takes as input two parameters: the period p (in days) on which to perform the linear regression, and the threshold l of the standardised spread that would activate the trading signals. In particular:

- $Z(Y - \beta X) > l$: LONG stock X, SHORT stock Y,
- $Z(Y - \beta X) < -l$: LONG stock Y, SHORT stock X,

where $Z(\cdot)$ is the function standardizing the spread. LONG means buying the stock that we currently deem undervalued, and SHORT means selling the stock that we consider overvalued, thus benefiting from a price increase of the first and a price decrease of the second, since we expect their spread will converge to the mean, which attains value 0 after performing the normalization.

What is the connection between GLASSO and pairs trading? The former can identify the best pairs of securities to be used in the latter; indeed, not every pair of securities is suitable for a pairs trading strategy since its spread might not obey the mean-reversion assumption. We therefore run the trading strategy only on the pairs that are connected in the GLASSO graph.

3.1 Model Selection

We can now find the GLASSO model that yields the optimal trading strategy for given parameters p and l. We fix those parameters because they deal with factors that we do not take into account in this analysis, such as the investment horizon and the risk aversion of the investor. Therefore, we do not aim at finding the best trading strategy overall, but rather the best GLASSO model, that is the best ρ for a given trading strategy. The model should seek a great risk-reward ratio, and bigger models (with more edges) should be penalized as trading more pairs implies incurring in higher fees. We define our proprietary optimality criteria as:

$$\frac{\bar{r}_t}{SD(r_t)} \cdot \sqrt{252 \cdot 3} - \lambda \cdot pen(PC),$$

where $\bar{r}_t$ is the average of the portfolio returns (proxy for reward), $SD(r_t)$ is their standard deviation (proxy for risk), λ is the penalization parameter and $pen(PC)$ is the number of non-zero correlations. We multiply by the square root factor to annualise the ratio over 3 years. It is important to note that portfolio returns are weighted according to the partial correlation of each pair, that is pairs that are strongly correlated have an higher weight in the calculation of portfolio returns. We run the strategy on the period 2017-2019 with $p = 180$ and $l = 0.2$, and by optimizing for the aforementioned criteria, we get that the optimal rho is 0.58 for $\lambda = 0.001$, 0.63 for $\lambda = 0.01$, and 0.68 for $\lambda = 0.05$.

3.2 Strategy Test

We perform an hypothesis test to check whether the strategy's returns are significantly higher than the ones of a random strategy, i.e. a strategy that selects random pairs. We run the strategy with $\rho = 0.58$, and compare it with 100 random strategies, each of which selects the same number of pairs as our strategy, in this case 55. Since mean-reversion strategies are negatively skewed (quantifiedstrategies.com, 2022), their returns are not normal and we cannot compare their distributions

with a two-sample t-test. For this reason, we resort to distribution-free two-sample tests, namely, Wilcoxon signed-rank test and Kolmogorov-Smirnov (KS). They are one-sided, with null hypothesis that the distribution of the GLASSO-based strategy has a lower or equal location compared to the random strategy.

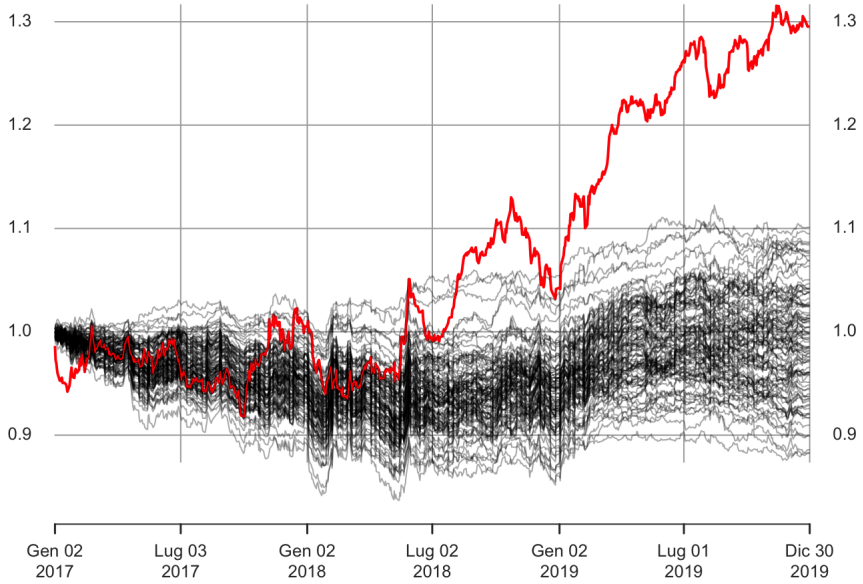

Figure 5: Cumulative return of GLASSO-based strategy (red) vs. random strategies (grey).

By running the two tests with $\alpha = 0.05$, we get that the KS test always rejects the null hypothesis, while Wilcoxon does not reject it only for one simulation out of 100. We can therefore conclude that our strategy delivers better risk-adjusted returns than the random one.

4 Limitations and Possible Improvements

Although we achieved interesting results, this analysis can be made more comprehensive. From a statistical point of view, it would be interesting to observe how the hypothesis tests change after introducing multiple testing corrections. Moreover, different model selection methods might be employed, such as the *step-down* method, which in our case would progressively remove pairs that do not significantly increase the risk-adjusted returns of the portfolio. Finally, from a financial point of view, it would be appropriate to consider excess returns, that is returns subtracted by the risk-free rate, and to take into account transaction costs, which in the case of high-frequency strategies can have a drastic impact on the performance.

5 Bibliography

- Ahad et al, 2011. "Sensitivity of Normality Tests to Non-normal Data".
- Mota, 2012. "Normality Assumption for the Log-return of the Stock Prices".
- Robot Wealth, 2020. "The Graphical Lasso and its Financial Applications".
- quantifiedstrategies.com, 2022. "Negatively Skewed Distribution in Trading Strategies?".

6 Appendix: R Code

```r
library(readr)
library(car)
library(tseries)
library(goftest)
library(dplyr)
library(xlsx)
library(lubridate)

file.choose()

#add specific date-related columns
dates <- as.POSIXct(stocks_prices$Date, format="%Y-%m-%d")
stocks_prices$year = format(dates, format="%Y")
stocks_prices$day_month = format(dates, format="%d")
stocks_prices$day_week = wday(stocks_prices$Date, week_start=1)
stocks_prices <- stocks_prices[,!names(stocks_prices) %in% c("day")]

stocks_names_columns <- setdiff(names(stocks_prices), c("Date", "year", "day_month", "day_week"))

alpha = 0.01

#matrix in which we save normality results
summary_results <- matrix(nrow = 0, ncol = 4)
colnames(summary_results) <- c("Year", "N", "Normality % (daily)", "Normality % (weekly)")

for (year_filter in 2014:2021) {
  prices_year = subset(stocks_prices, year==year_filter)
  normality_perc <- c()
  #we check both daily and weekly returns
  for (lag_period in c(1, 7)) {
    normality_count <- 0
    stocks_count <- 0
    #with weekly returns we get only last trading day of the week (friday)
    if (lag_period == 7) {
      prices_year <- subset(prices_year, day_week==5)
    }
    for (column in stocks_names_columns) {
      prices = as.matrix(prices_year[ , column])
      #apply log return transformation
      log_returns <- diff(log(prices), lag=1)
      len <- sum(!is.na(log_returns))
      #assumption on the sample size of Shapiro-Wilk test
      if (len > 3 & len < 5000) {
        #Shapiro-Wilk test
        p_value <- shapiro.test(log_returns)$p.value
        normality <- p_value > alpha
        if (normality) {
          #count stocks that follow a normal distribution
          normality_count <- normality_count + 1
        }
        stocks_count <- stocks_count + 1
      }
    }
    normality_perc <- append(normality_perc, normality_count/stocks_count*100)
  }
  summary_results <- rbind(
    summary_results,
    c(
      year_filter,
      stocks_count,
      normality_perc
    )
  )
}

write.xlsx(as.matrix(summary_results), "/Users/fede/Documents/Assigment MS/Graphical
Lasso/Code/normality_shapiro.xlsx")

#QQPlot showed in the paper
qqPlot(diff(log(subset(stocks_prices, day_week==5 & year==2015)$MONC), lag=1), ylab="MONC weekly log-returns")
```

Figure 6: Code to conduct normality tests on log returns.

```r
library(glasso)
library(igraph)
library(threejs)
library(htmlwidgets)
library(lubridate)

#calculate the partial correlations matrix
calculate_partial_correlations = function(stocks_prices, rho, start_period, end_period) {
  #add specific date-related columns
  stocks_prices$year = year(stocks_prices$Date)
  stocks_prices$day_week = wday(stocks_prices$Date, week_start=1)
  #we filter for a specific period of time, and we get only last day of the week (weekly returns)
  stocks_prices = subset(stocks_prices, year>=start_period & year<=end_period & day_week==5)
  stocks_names_columns = setdiff(names(stocks_prices), c("Date", "year"))
  stocks_prices = stocks_prices[, stocks_names_columns]
  #filter null values
  stocks_prices = stocks_prices[,colSums(is.na(stocks_prices))<nrow(stocks_prices)]
  #apply log return transformation
  stocks_log_returns = diff(log(as.matrix(stocks_prices)), lag=1)
  #apply normalization
  stocks_log_returns = scale(stocks_log_returns)
  #calculate covariance matrix
  cov_matrix = var(stocks_log_returns)
  #filter null values
  cov_matrix = cov_matrix[,colSums(is.na(cov_matrix))<nrow(cov_matrix)]
  cov_matrix = na.omit(cov_matrix)
  #apply GLASSO, that retrieves the precision matrix
  precision_matrix = glasso(cov_matrix, rho=rho, approx=FALSE)$wi
  colnames(precision_matrix) = colnames(cov_matrix)
  rownames(precision_matrix) = rownames(cov_matrix)
  #make sure that the precision matrix is symmetric
  if(!isSymmetric(precision_matrix)) {
    precision_matrix[lower.tri(precision_matrix)] = t(precision_matrix)[lower.tri(precision_matrix)]
  }
  #calculate partial correlations matrix with the formula presented in the paper
  parr.corr = matrix(nrow=nrow(precision_matrix), ncol=ncol(precision_matrix))
  for(k in 1:nrow(parr.corr)) {
    for(j in 1:ncol(parr.corr)) {
      parr.corr[j, k] = -precision_matrix[j,k]/sqrt(precision_matrix[j,j]*precision_matrix[k,k])
    }
  }
  colnames(parr.corr) = colnames(precision_matrix)
  rownames(parr.corr) = colnames(precision_matrix)
  #set the diagonal elements (variances) to 0
  diag(parr.corr) = 0
  return (parr.corr)
}

#retrieves the name of the pairs and the strength of their correlation
retrieve_pairs = function(partial_corr) {
  pairs_matrix = matrix(nrow = 0, ncol = 3)
  colnames(pairs_matrix) = c("Stock1", "Stock2", "Correlation")
  for(k in 1:nrow(partial_corr)) {
    for(j in 1:ncol(partial_corr)) {
      if (partial_corr[j, k] != 0) {
        values = sort(c(rownames(partial_corr)[k], colnames(partial_corr)[j]))
        values = append(values, partial_corr[j, k])
        pairs_matrix = rbind(
          pairs_matrix,
          values
        )
      }
    }
  }
  return(unique(pairs_matrix))
}

#draws the chart presented in the paper (rho vs. number of graph edges)
stocks_prices = read.csv("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/stocks_prices.csv")
rho_array = seq(0.01, 1, by=0.01)
connections_array = c()
for (rho in rho_array) {
  parr_corr = calculate_partial_correlations(stocks_prices, rho, 2014, 2016)
  connections_array = c(connections_array, sum(colSums(parr_corr != 0))/2)
}
plot(rho_array, connections_array, type="l", xlab="Rho", ylab="Edges")'
```

Figure 7: Code to create a GLASSO model.

```r
library(PairTrading)
library(PerformanceAnalytics)
library(lubridate)

#given a pair of stocks, we calculate their return with the PairTrading package
calculate_pair_return = function (stocks_prices, pair, period, signal_threshold) {
  price.pair = xts(stocks_prices[,pair], order.by = as.Date(stocks_prices$Date))
  price.pair = na.omit(price.pair)
  try = tryCatch(
    {
      #functions of the PairTrading package
      params = EstimateParametersHistorically(price.pair, period = period)
      signal = Simple(params$spread, spread.entry = signal_threshold)
      return.pairtrading = PairTrading::Return(price.pair, stats::lag(signal), stats::lag(params$hedge.ratio))
      return(return.pairtrading)
    }, error = function(cond) {
      return (-1)
    }
  )
  return(try)
}

calculate_portfolio_return = function(stocks_prices, pairs_matrix, end_glasso, strategy_duration, period,
signal_threshold, random=FALSE) {
  stocks_prices$year = year(stocks_prices$Date)
  stocks_prices = subset(stocks_prices, year>=end_glasso & year<=end_glasso+strategy_duration)
  portfolio_return = data.frame()
  #file in which we already saved some returns, used to speed up the process
  saved_portfolio_returns = read.csv("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/saved_returns.csv")
  #array of weights of the pairs
  weights = c()
  for(i in 1:nrow(pairs_matrix)) {
    pair = c(pairs_matrix[i,1], pairs_matrix[i,2])
    pair_name = paste(pairs_matrix[i,1], pairs_matrix[i,2], sep = "_")
    #checks if we already saved the return in the file
    if (pair_name %in% colnames(saved_portfolio_returns)) {
      pair_return = data.frame(saved_portfolio_returns[pair_name])
      pair_return$Date = as.Date(as.vector(unlist(saved_portfolio_returns[1])), format="%Y-%m-%d")
    } else {
      pair_return = data.frame(calculate_pair_return(stocks_prices, pair, period, signal_threshold))
      if (length(rownames(pair_return)) > 1) {
        pair_return$Date = as.Date(rownames(pair_return))
      }
    }
    if (length(rownames(pair_return)) > 1) {
      colnames(pair_return)[1] = pair_name
      #assigns a weight to each pair which is the partial correlation
      if (!random) {
        weights = append(weights, abs(as.double(pairs_matrix[i,3])))
      }
      if (length(rownames(portfolio_return)) == 0) {
        portfolio_return = pair_return
      } else {
        #merges the returns of each pair
        portfolio_return = merge(portfolio_return, pair_return, by = "Date", all = TRUE)
      }
    }
  }
  rownames(portfolio_return) = portfolio_return$Date
  portfolio_return$year = year(portfolio_return$Date)
  #avoid intersection between GLASSO period and strategy period (as with training and test)
  portfolio_return = subset(portfolio_return, year>end_glasso & year<=end_glasso+strategy_duration)
  portfolio_return = portfolio_return[-1]
  portfolio_return = portfolio_return[-ncol(portfolio_return)]
  #if we use random pairs, we do not assign weights to each pair
  if (random) {
    return(PerformanceAnalytics::Return.portfolio(portfolio_return, rebalance_on = "months"))
  } else {
    return(PerformanceAnalytics::Return.portfolio(portfolio_return, weights = weights, rebalance_on = "months"))
  }
}

#calculates cumulative return
calculate_cum_return = function(returns) {
  return(cumprod(1 + returns))
}
```

Figure 8: Code to perform the pairs trading strategy on a set of pairs.

```r
source("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/pairs_trading_with_library.R")
source("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/GLASSO.R")

start_glasso = 2014
end_glasso = 2016
#the strategy period is 2017-2019 (3 years)
strategy_duration = 3
#p in the paper
period = 180
#called l in the paper
signal_threshold = 0.2
lambda = 0.05

stocks_prices = read.csv("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/stocks_prices.csv")

#calculates the GLASSO-based strategy portfolio return for a given rho
calculate_strategy_return = function(rho) {
  parr_corr = calculate_partial_correlations(stocks_prices, rho, start_glasso, end_glasso)
  pairs_matrix = retrieve_pairs(parr_corr)
  portfolio_return_strategy = calculate_portfolio_return(stocks_prices, pairs_matrix, end_glasso, strategy_duration,
period, signal_threshold)
  return(portfolio_return_strategy)
}

#function to be maximised during the model selection
model_selection_criteria = function (rho) {
  portfolio_return_strategy = calculate_strategy_return(rho)
  cum_return_strategy = calculate_cum_return(portfolio_return_strategy)
  final_return = cum_return_strategy[length(cum_return_strategy)]
  connections = sum(colSums(calculate_partial_correlations(stocks_prices, rho, start_glasso, end_glasso) != 0)) / 2
  #calculates the optimal criteria, as defined in the paper
  optimal_criteria = mean(portfolio_return_strategy) / sd(portfolio_return_strategy) * sqrt(252 * strategy_duration)
- lambda * connections
  return(optimal_criteria)
}

#find the optimal rho for given trading strategy parameters
#rho is bound to be less than 0.82 because above that threshold there are no pairs
optimise(model_selection_criteria, interval = c(0, 0.82), tol = 0.01, maximum = TRUE)
```

Figure 9: Code for model selection.

```r
source("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/pairs_trading_with_library.R")
source("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/model_selection.R")

stocks_prices = read.csv("/Users/fede/Documents/Assigment MS/Graphical Lasso/Code/stocks_prices.csv")

end_glasso = 2016
strategy_duration = 3
period = 180
signal_threshold = 0.2

#optimal rho with lambda = 0.01
optimal_rho = 0.58
#number of connections with lambda = 0.01 (used to generate the random pairs)
optimal_n = 55

#calculates GLASSO-based strategy return
portfolio_return_strategy = calculate_strategy_return(optimal_rho)
cum_return_strategy = calculate_cum_return(portfolio_return_strategy)

#same function as retrieve_pairs but with n random pairs instead
generate_random_pairs_matrix = function(n) {
  stocks_names_columns = setdiff(names(stocks_prices), c("Date", "year"))
  combinations = t(combn(stocks_names_columns, 2))
  #replace = FALSE because we can't repeat pairs
  random_indices = sample(1:dim(combinations)[1], n, replace = FALSE)
  return(combinations[random_indices,])
}

#helper function used just to compare two strategies' returns
merge_portfolios = function(portfolio_return_strategy, portfolio_return_random) {
  colnames(portfolio_return_strategy)[1] = "portfolio_return_strategy"
  portfolio_return_strategy$Date = as.Date(rownames(portfolio_return_strategy))
  colnames(portfolio_return_random)[1] = "portfolio_return_random"
  portfolio_return_random$Date = as.Date(rownames(portfolio_return_random))
  return(merge(portfolio_return_strategy, portfolio_return_random, by = "Date", all = FALSE))
}

plot(cum_return_strategy, col="red")

alpha = 0.05
#variables to count the number of times we reject the null hypothesis
passed_wilcox = 0
passed_ks = 0

for (i in 1:100) {
  random_pairs_matrix = generate_random_pairs_matrix(optimal_n)
  tryCatch(
    expr = {
      random_returns = calculate_portfolio_return(stocks_prices, random_pairs_matrix, end_glasso, strategy_duration,
period, signal_threshold, TRUE)
      portfolios = merge_portfolios(portfolio_return_strategy, random_returns)
      #Wilcoxon signed-rank test
      p_value_wilcox = wilcox.test(as.vector(unlist(portfolios$portfolio_return_strategy)),
as.vector(unlist(portfolios$portfolio_return_random)), paired = TRUE, alternative = "g")$p.value
      #Kolmogorov-Smirnov test
      p_value_ks = ks.test(as.vector(unlist(portfolios$portfolio_return_strategy)),
as.vector(unlist(portfolios$portfolio_return_random)), alternative = "g")$p.value
      cum_return_random = calculate_cum_return(portfolios$portfolio_return_random)
      lines(cum_return_random, col=rgb(0, 0, 0, 0.35))
      if (p_value_wilcox < alpha) {
        passed_wilcox = passed_wilcox + 1
      }
      if (p_value_ks < alpha) {
        passed_ks = passed_ks + 1
      }
    }
  )
}

#trick to update the graph, otherwise it doesn't show the new lines
lines(calculate_cum_return(portfolio_return_strategy), col="red")
#print the results of the hypothesis tests
print(c(passed_wilcox, passed_ks))
```

Figure 10: Code to compare the GLASSO-based strategy with the random one.