Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

TASK 2

DSAF

Martí Farré
Sergio Cárdenas
Anna Ramon

February 10th, 2024

Problem 1

Backtesting trading strategies based on sentiment indicators. Consider the dataset *dataSent.zip* provided in Atenea/Home Work 2 containing the stock price history and associated sentiment indicators for various stocks. Select two and register their tickers in the Atenea/spreadsheet in column HW2.1. Try to select different pair from those selected by others (not mandatory). Use the R script *Rlab6_TradingStrategywCost.R* which builds the following compositions of sentiment indicators:

$$BULL = positiveP\ artscr + certaintyP\ artscr + f\ inupP\ artscore$$

$$BEAR = negativeP\ artscr + uncertaintyP\ artscr + f\ indownP\ artscore$$

$$BBr = 100 \cdot BULL/(BULL + BEAR)\ with\ NAs\ interpolation$$

$$PNlog = 0.5 \cdot log((positiveP + 1)/(negativeP + 1))$$

and defines performance measures and a trading strategy based on Moving Average crossover (MAcross) applied to some explanatory variable (ev). There are two test functions: one for trading the whole period, other for a rolling window analysis over fixed length periods (window-size= 1 yr, 6 mon).
Your team's task is to do the following with your selected stocks:

- **Design one trading strategy (i.e. the signaling sequence) different from the MAcross strategy shown in Lecture 6, using any (or all) sentimental indicators ev in positiveP, negativeP, BULL, BEAR, BBr, P Nlog, and RV T (volume of news) and test performance of your strategy for the cases where you either allow long-only or long-short trading, and using the rolling window analysis with window sizes = 254 (1 year) or 127 (6 months). Any other parameters of your strategies must be tuned.**
  (Suggestions: adapt some of the many strategies from Technical Analysis to the sentimental indicators ev.
  See https://www.avatrade.com/education/technical-analysis-indicators-strategies; other examples in https://rpubs.com/johnakwei/207852, or explore the web for technical analysis trading strategies.
  You may also combine your sentiment-based strategy with other indicators based on price, e.g. lags, volatility...)

In all cases report successful results (where excess return Me —BH is positive).

In this exercise, we have designed a trading strategy based on the BBr sentimental indicator, with formula:

$$BBr = 100 \cdot BULL/(BULL + BEAR)$$

Furthermore, we have decided to use RVT, which represents the volume of news. With that, and using a threshold for BBr and another one for RVT, we can define our strategy:

- If BBr and RVT are above their respective threshold, buy.
- If BBr is below the inverse of its threshold, sell.
- If none of the above conditions are met, don't do anything if it is a long-only signal and sell if it is a long and short one.

The following code defines a test function for this strategy:

```
testBBrStrategy <- function(myStock, BBr, RVT, bbr_threshold = 1, rvt_
    threshold = 0, longshort = 0, tcost = 0) {

  # Generate signals based on Bull-Bear ratio threshold
  myPosition <- sig <- Lag(ifelse(BBr > bbr_threshold & RVT > rvt_threshold
      , 1, ifelse(BBr < (1/bbr_threshold), -1, longshort)), 1)

  # Calculate returns
  bmkReturns <- dailyReturn(myStock, type = "arithmetic")
  myReturns <- bmkReturns * myPosition

  # Rename columns
  names(bmkReturns) <- 'BH'
  names(myReturns) <- 'BBr'

  # Merge returns and remove NA values
  tt <- na.omit(merge(bmkReturns, myReturns))

  # Calculate performance
  cbind(BBr = Performance(tt$BBr, cost = tcost), BH = Performance(tt$BH, 2,
      tcost))
}
```

Also, as the exercise required, we have implemented the following code for testing the strategy using a rolling window:

```
RollingTestBBrStrategy <- function(myStock, BBr, RVT, bbr_threshold = 1,
    rvt_threshold = 0, longshort = 0, w_size = 252) {
  # Generate signals based on Bull-Bear ratio threshold
  myPosition <- sig <- Lag(ifelse(BBr > bbr_threshold & RVT > rvt_threshold
      , 1, ifelse(BBr < (1/bbr_threshold), -1, longshort)), 1)

  # Calculate returns
  bmkReturns <- dailyReturn(myStock, type = "arithmetic")
  myReturns <- bmkReturns * myPosition

  # Rename columns
  names(bmkReturns) <- 'BH'
  names(myReturns) <- 'BBr'

  # Merge returns and remove NA values
  tt <- na.omit(merge(bmkReturns, myReturns))

  n_windows = nrow(tt) - w_size
  if (n_windows < 1) stop("Window size too large")

  # Perform rolling window analysis
  perform = foreach(i = 1:n_windows, .combine = rbind) %do% {
    bhStra = tt$BH[i:(w_size + i - 1), ]
    bbrStra = tt$BBr[i:(w_size + i - 1), ]
    per = rbind(BH = Performance(bhStra), BBr = Performance(bbrStra))
    return(per)
  }

  bhindx = seq(1, 2 * n_windows, 2)
  bbrindx = seq(2, 2 * n_windows, 2)
  BHmeans = colMeans(perform[bhindx, ])
```

```
  BBrMeans = colMeans(perform[bbrindx, ])
  MeanPerf = rbind(BHmeans, BBrMeans)
  colnames(MeanPerf) = colnames(perform)
  rownames(MeanPerf) = c("BH", "BBr")

  return(list("AvgPerf" = MeanPerf, "NumWindows" = n_windows))
}
```

With that, we can test our strategy on two different stocks. We have decided to choose *Google (GOOG)* and *Microsoft (MSFT)*. For doing that, we first defined some hyper parameter values, as shown in the following code:

```
bbr_threshold_values <- list(0.9, 1, 1.1)
rvt_threshold_values <- list(0, 0.0005, 0.001)
longshort_values <- list(0, -1)
tcost_values <- list(0, 0.01, 0.05)
w_size_values <- list(254, 254/2)
```

So, now we can run the test. Take into account that the *Google* and *Microsoft* stocks and sentimental indicators have been previously extracted. This first code shows the Google full period test:

```
for (i in seq_along(bbr_threshold_values)) {
  for (j in seq_along(rvt_threshold_values)) {
    for (k in seq_along(longshort_values)) {
      for (l in seq_along(tcost_values)) {
        res <- testBBrStrategy(goog_target, goog_BBr, goog_RVT, bbr_
            threshold = bbr_threshold_values[[i]], rvt_threshold = rvt_
            threshold_values[[j]], longshort = longshort_values[[k]], tcost
             = tcost_values[[l]])
        if(res[1, 1]-res[1, 2]>0) print(paste("BBr threshold =", bbr_
            threshold_values[[i]], "; RVT threshold =", rvt_threshold_
            values[[j]], "; longshort =", longshort_values[[k]], "; tcost =
            ", tcost_values[[l]]))
        if(res[1, 1]-res[1, 2]>0) print(paste("Excess return =", res[1, 1]-
            res[1, 2]))
      }
    }
  }
}
```

As you can see, we will print the excess return of the iterations where our strategy outperforms a buy-and-hold one. That is, the successful runs, where the excess return is positive. After running it, we obtain the following output:

```
"BBr threshold = 5 ; RVT threshold = 0 ; longshort = 0 ; tcost = 0.05"
"Excess return = 0.0136124940252244"
```

As we can observe, the only successsful runs was the one with a BBr threshold of 5, a RVT threshold of 0, and a tcost of 0.05 for a long-only signal. Also, the excess return is not quite high, so our strategy seems to work not too good for this case, which could make sense due to its simplicity. To confirm our thoughts, we run this experiment for the *Microsoft* case, using the following code:

```
for (i in seq_along(bbr_threshold_values)) {
  for (j in seq_along(rvt_threshold_values)) {
```

```
    for (k in seq_along(longshort_values)) {
      for (l in seq_along(tcost_values)) {
        res <- testBBrStrategy(msft_target, msft_BBr, msft_RVT, bbr_
            threshold = bbr_threshold_values[[i]], rvt_threshold = rvt_
            threshold_values[[j]], longshort = longshort_values[[k]], tcost
             = tcost_values[[l]])
        if(res[1, 1]-res[1, 2]>0) print(paste("BBr threshold =", bbr_
            threshold_values[[i]], "; RVT threshold =", rvt_threshold_
            values[[j]], "; longshort =", longshort_values[[k]], "; tcost =
            ", tcost_values[[l]]))
        if(res[1, 1]-res[1, 2]>0) print(paste("Excess return =", res[1, 1]-
            res[1, 2]))
      }
    }
  }
}
```

The output obtained after executing it is:

```
"BBr threshold = 5 ; RVT threshold = 0 ; longshort = 0 ; tcost = 0.05"
"Excess return = 0.0275193399856859"
"BBr threshold = 5 ; RVT threshold = 0 ; longshort = -1 ; tcost = 0.05"
"Excess return = 0.0284808790634858"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; tcost = 0"
"Excess return = 0.0799232701279631"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; tcost = 0.01"
"Excess return = 0.0899232701279631"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; tcost = 0.05"
"Excess return = 0.129923270127963"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; tcost = 0"
"Excess return = 0.18360831086383"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; tcost = 0.01"
"Excess return = 0.19360831086383"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; tcost = 0.05"
"Excess return = 0.23360831086383"
```

In that case, the results were better than for the *Google* one. We can observe that 0 is a value for the RVT threshold that is giving good results, combined with a BBr threshold of 25. In general, we can see that the excess return values are quite higher than before, so we could say that despite its simplicity, this strategy is working decently for the *Microsoft* stock.

Now, we want to test the strategy for both stocks using a rolling window with the code defined earlier. For the *Google* case, we will use the following code:

```
for (i in seq_along(bbr_threshold_values)) {
  for (j in seq_along(rvt_threshold_values)) {
    for (k in seq_along(longshort_values)) {
      for (l in seq_along(w_size_values)) {
        meanperf <- RollingTestBBrStrategy(goog_target, goog_BBr, goog_RVT,
             bbr_threshold = bbr_threshold_values[[i]], rvt_threshold = rvt
            _threshold_values[[j]], longshort = longshort_values[[k]], w_
            size = w_size_values[[l]])
        if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
            BBr threshold =", bbr_threshold_values[[i]], "; RVT threshold =
            ", rvt_threshold_values[[j]], "; longshort =", longshort_values
            [[k]], "; Window Size =", w_size_values[[l]]))
        if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
            Excess return =", meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1,
```

```
            1]))
      }
    }
  }
}
```

When executing it, we obtain the following output:

```
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; Window Size = 254
    "
"Excess return = 0.0148390840628059"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; Window Size = 127
    "
"Excess return = 0.00582246732208546"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; Window Size =
    254"
"Excess return = 0.0290361468692051"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; Window Size =
    127"
"Excess return = 0.0116115356485468"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; longshort = 0 ; Window Size =
     254"
"Excess return = 0.00180420199994967"
```

Now, the results are better than for the full period test for the *Google* stock, with some decent excess return values for hyper parameter combinations that gave us good results in previous tests.

With the following code we can test this experiment for the *Microsoft* stock:

```
for (i in seq_along(bbr_threshold_values)) {
  for (j in seq_along(rvt_threshold_values)) {
    for (k in seq_along(longshort_values)) {
      for (l in seq_along(w_size_values)) {
        meanperf <- RollingTestBBrStrategy(msft_target, msft_BBr, msft_RVT,
             bbr_threshold = bbr_threshold_values[[i]], rvt_threshold = rvt
            _threshold_values[[j]], longshort = longshort_values[[k]], w_
            size = w_size_values[[l]])
        if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
            BBr threshold =", bbr_threshold_values[[i]], "; RVT threshold =
            ", rvt_threshold_values[[j]], "; longshort =", longshort_values
            [[k]], "; Window Size =", w_size_values[[l]]))
        if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
            Excess return =", meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1,
            1]))
      }
    }
  }
}
```

Now, we execute the code, and we obtain the following results:

```
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; Window Size = 254
    "
"Excess return = 0.0462346822541656"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = 0 ; Window Size = 127
    "
"Excess return = 0.0133627424897269"
```

```
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; Window Size = 254"
"Excess return = 0.0996397455655098"
"BBr threshold = 25 ; RVT threshold = 0 ; longshort = -1 ; Window Size = 127"
"Excess return = 0.0304175787924255"
```

In that case, these results are worse than the ones obtained without the use of the rolling window. But, even in that case, the hyper parameter combination that gives better results is the same than before.

So, we could conclude that for the *Google* case, the use of a rolling window has helped on improving our strategy capabilities, while on the *Microsoft* case, the use of a rolling window is not improving the performance. In any case, we have managed to obtain quite decent results for each stock with our strategy.

However, we can try to improve the performance by combining our strategy with other indicators. We have decided to use the volatility. So, we have modify our rolling test strategy in order to also depend on volatility. This is done by adding the condition that the volatility should be higher than a certain threshold to buy. The following code shows this modification:

```
RollingTestBBrVolStrategy <- function(myStock, BBr, RVT, volatility, bbr_threshold = 1, rvt_threshold = 0, volatility_threshold = 0, longshort = 0, w_size = 252) {
  # Generate signals based on Bull-Bear ratio threshold
  myPosition <- sig <- Lag(ifelse(BBr > bbr_threshold & RVT > rvt_threshold & volatility > volatility_threshold, 1, ifelse(BBr < (1/bbr_threshold), -1, longshort)), 1)

  # Calculate returns
  bmkReturns <- dailyReturn(myStock, type = "arithmetic")
  myReturns <- bmkReturns * myPosition

  # Rename columns
  names(bmkReturns) <- 'BH'
  names(myReturns) <- 'BBr'

  # Merge returns and remove NA values
  tt <- na.omit(merge(bmkReturns, myReturns))

  n_windows = nrow(tt) - w_size
  if (n_windows < 1) stop("Window size too large")

  # Perform rolling window analysis
  perform = foreach(i = 1:n_windows, .combine = rbind) %do% {
    bhStra = tt$BH[i:(w_size + i - 1), ]
    bbrStra = tt$BBr[i:(w_size + i - 1), ]
    per = rbind(BH = Performance(bhStra), BBr = Performance(bbrStra))
    return(per)
  }

  bhindx = seq(1, 2 * n_windows, 2)
  bbrindx = seq(2, 2 * n_windows, 2)
  BHmeans = colMeans(perform[bhindx, ])
  BBrMeans = colMeans(perform[bbrindx, ])
  MeanPerf = rbind(BHmeans, BBrMeans)
  colnames(MeanPerf) = colnames(perform)
  rownames(MeanPerf) = c("BH", "BBr")
```

```
34
35    return(list("AvgPerf" = MeanPerf, "NumWindows" = n_windows))
36  }
```

In that case, we are testing this function just for the *Google* stocks as it was the one that obtained better results for the rolling window approach. The following code runs the experiment:

```
1  for (i in seq_along(bbr_threshold_values)) {
2    for (j in seq_along(rvt_threshold_values)) {
3      for (k in seq_along(volatility_threshold_values)) {
4        for (l in seq_along(longshort_values)) {
5          for (m in seq_along(w_size_values)) {
6            meanperf <- RollingTestBBrVolStrategy(goog_target, goog_BBr, goog
                 _RVT, goog_volatility, bbr_threshold = bbr_threshold_values[[
                 i]], rvt_threshold = rvt_threshold_values[[j]], volatility_
                 threshold = volatility_threshold_values[[k]], longshort =
                 longshort_values[[l]], w_size = w_size_values[[m]])
7            if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
                 BBr threshold =", bbr_threshold_values[[i]], "; RVT threshold
                  =", rvt_threshold_values[[j]], "; Volatility threshold =",
                 volatility_threshold_values[[k]], "; longshort =", longshort_
                 values[[l]], "; Window Size =", w_size_values[[m]]))
8            if(meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1, 1]>0) print(paste("
                 Excess return =", meanperf$AvgPerf[2, 1]-meanperf$AvgPerf[1,
                 1]))
9          }
10       }
11     }
12   }
13 }
```

When executing this code, we obtained the following results:

```
1  "BBr threshold = 5 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
       longshort = 0 ; Window Size = 254"
2  "Excess return = 0.067715074396004"
3  "BBr threshold = 5 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
       longshort = 0 ; Window Size = 127"
4  "Excess return = 0.0223051385782107"
5  "BBr threshold = 5 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
       longshort = -1 ; Window Size = 254"
6  "Excess return = 0.135334308956105"
7  "BBr threshold = 5 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
       longshort = -1 ; Window Size = 127"
8  "Excess return = 0.0446811729719563"
9  "BBr threshold = 5 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
       longshort = 0 ; Window Size = 254"
10 "Excess return = 0.0668839349276919"
11 "BBr threshold = 5 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
       longshort = 0 ; Window Size = 127"
12 "Excess return = 0.0197672507985516"
13 "BBr threshold = 5 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
       longshort = -1 ; Window Size = 254"
14 "Excess return = 0.116401498286609"
15 "BBr threshold = 5 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
       longshort = -1 ; Window Size = 127"
16 "Excess return = 0.0327541462831222"
17 "BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0 ;
```

```
    longshort = 0 ; Window Size = 254"
"Excess return = 0.0136585649500426"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0 ;
    longshort = 0 ; Window Size = 127"
"Excess return = 0.00490034365075524"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0 ;
    longshort = -1 ; Window Size = 254"
"Excess return = 0.0266860604355579"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0 ;
    longshort = -1 ; Window Size = 127"
"Excess return = 0.00980461094063408"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
    longshort = 0 ; Window Size = 254"
"Excess return = 0.0857779457555977"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
    longshort = 0 ; Window Size = 127"
"Excess return = 0.0294124233360825"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
    longshort = -1 ; Window Size = 254"
"Excess return = 0.1730724846343"
"BBr threshold = 25 ; RVT threshold = 0 ; Volatility threshold = 0.1 ;
    longshort = -1 ; Window Size = 127"
"Excess return = 0.059249121246445"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; Volatility threshold = 0 ;
    longshort = 0 ; Window Size = 254"
"Excess return = 0.00110693412405724"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
    longshort = 0 ; Window Size = 254"
"Excess return = 0.0738406917387581"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
    longshort = 0 ; Window Size = 127"
"Excess return = 0.0226275539508611"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
    longshort = -1 ; Window Size = 254"
"Excess return = 0.130170245523847"
"BBr threshold = 25 ; RVT threshold = 5e-04 ; Volatility threshold = 0.1 ;
    longshort = -1 ; Window Size = 127"
"Excess return = 0.0384952178111578"
```

As it can clearly be seen, now we are obtaining more successful runs, which means that our strategy is outperforming the buy-and-hold one more than before. This is a sign that the inclusion of volatility in our strategy has helped on improving its performance.

Problem 2

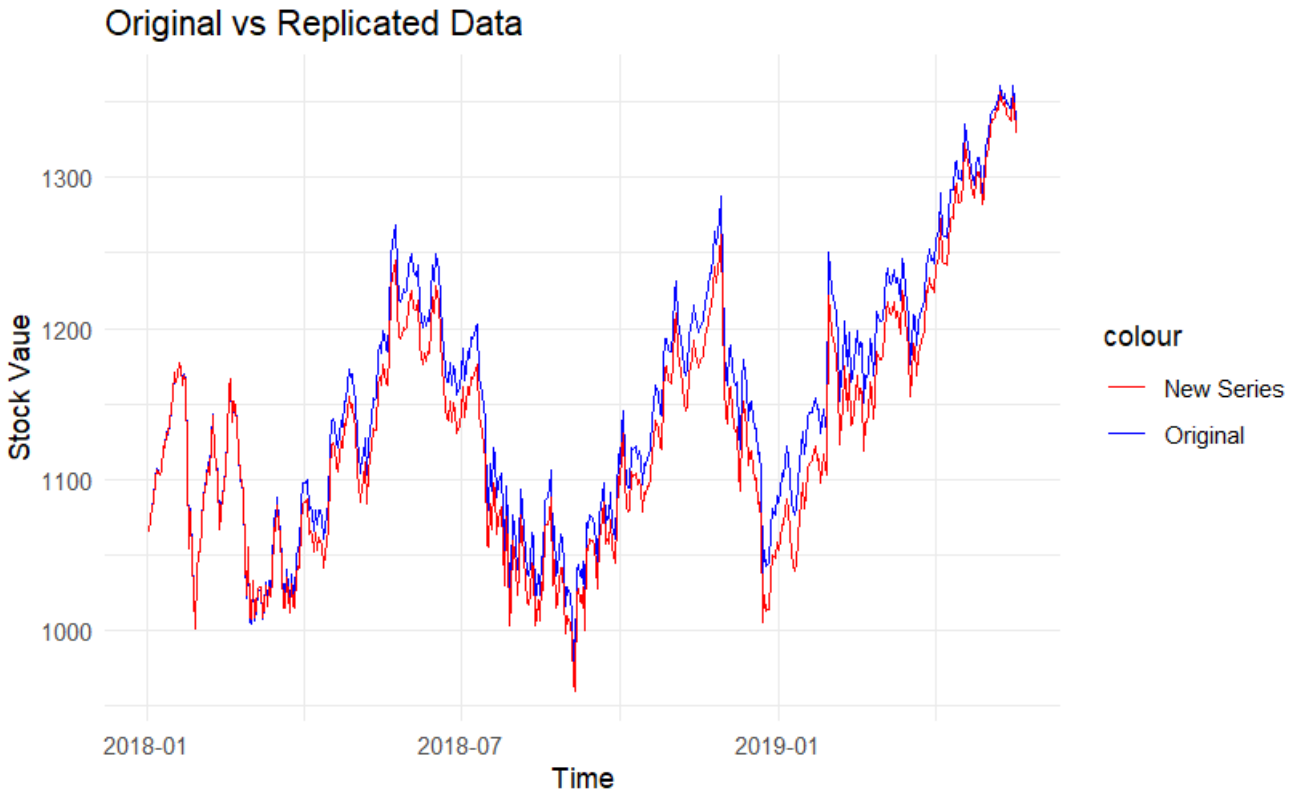
In the previous problem test your trading strategy with simulated data. This can be: a model that you fit to your stock prices (ARMA, GARCH, NNet, ...) or stationary bootstrap. Then generate various (e.g. 30) replicas of the stock price series and apply your strategy using corresponding original sentiment series. Report average excess return and number of successful runs.

In this exercise, our goal was to test the robustness of our trading strategy. To do so, we have created 30 replicas of our target (the adjusted closing stock value). We have fitted an ARIMA to model our time series, and the best found model it is later used to simulate similar time series.

```
set.seed(123)
```

```r
ts_target <- data[, 5]

n_replicas <- 30
n_periods <- length(ts_target)
target_replicas <- matrix(0, nrow = n_periods, ncol = n_replicas)

best_arima <- function(data){
  best_model <- list(order = c(0,0,0), aic = Inf)
  for(p in 0:3){
    for(d in 0:1){
      for(q in 0:3){
        model <- tryCatch(
          auto.arima(data, seasonal = FALSE, order = c(p,d,q), stepwise =
              FALSE),
          error = function(e) NULL
        )
        if (!is.null(model) && model$aic < best_model$aic){
          best_model$order <- model$order
          best_model$aic <- model$aic
        }
      }
    }
  }
  return(best_model)
}

arima_model <- best_arima(ts_target)

for(i in 1:n_replicas){
  replica <- arima.sim(list(order = arima_model$order), n = n_periods)
  target_replicas[, i] <- cumsum(replica) + ts_target
}
```

We can check if our replicas are similar to the original time series plotting them a sample replica with the real one:



We can see that our new time series behaves very similar to the original one.

The strategy that we will test is the one described on the previous exercise, with the better hyper parameter combination obtained, so the only modifications done have been when computing the strategy. Now, we do the testing 30 times, one for each new time series. We will store the successful runs and the average excess return.

```
excess_return <- c()
successful_run <- c()
##Full period test for each ev with longshort=0,-1, tcost=0,0.01,0.05
for(r in 1: n_replicas){
  up_target <- target_replicas[, r]
  new_data <- merge(data, up_target)
  target<-new_data$up_target
  names(target) <- "GOOG"
  res <- testBBrStrategy(target, BBr, RVT, bbr_threshold = 25, rvt_
      threshold = 0, longshort = 0, tcost = 0.05)
  excess_return <-c(excess_return, res[1] - res[9])
  if (res[1] - res[9] > 0) {
      successful_run <- c(successful_run, 1)
    } else {
      successful_run <- c(successful_run, 0)
    }
}
```

After running the test, we see that a 100% of our runs are successful and, on average, we obtain almost a 10% more return than our baseline. Therefore, we can confidently say that our testing strategy it is a successful one, as we are obtaining a larger return.

Problem 3

Choose 9 stocks from the dataset dataset.rds for Lec. 8 on Factor Models and Sentiment, and compute for these the Robust (ellipsoid) Global Maximum Return Portfolio using as perturbation matrix Sigma (S) the one corresponding to the factor models:1) the Fama-French 3-factors returns 3FF; and 2) the Sentiment indicator PNlog factor model. Use a two consecutive years of data for experiments selected from 2015-01 to 2020-06; register this period of data in Atenea/drive/spreadsheet column HW 2.2. Try to select different 2-years period from those selected by others (not mandatory). Try different kappas (2 or 3 values in (0, 1)) and multiple robust noisy solutions to check for sensitivity. Comment on the differences/similarities of results for both cases of Sigma.

In this exercise, we were asked to compute the Robust Global Maximum Return Portfolio using two different perturbation matrices Sigma (S): one corresponding to the Fama-French 3-factor model (3FF) and another corresponding to a Sentiment indicator PNlog factor model. The two-year period selected for performing the analysis is [2018/01/01 - 2020/12/20]. The goal is to use the estimated covariance of capital asset returns provided by the factor models for computing the optimal portfolio. Covariance matrices of asset returns are fundamental for choosing diversified portfolios and are key inputs to portfolio optimization routines.

The main purpose of this exercise is to optimize this problem:

$$\begin{aligned} \text{maximize} \quad & \mu^T w \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1 \\ & w_i \geq 0, \quad \forall i \end{aligned}$$

To obtain the global maximum return of a portfolio some algorithms tend to allocate all the budget on the single stock with largest expected return. This strategy leads to a constant change of the optimal allocation from realization to realization, which is highly undesirable. To prevent this assume μ belongs to some convex uncertainty set:

$$\mu' = \text{colMeans}(X) + k * S^{0.5} u, \quad \|u\| < 1,$$

The steps followed for performing the proposed task are:

1. Load the dataset *dataset.rds* for Lec. 8 and select 9 stocks:

```
dataset <- readRDS("C:/Users/HP/OneDrive/Escritorio/ANNA/UPC/DSAF/
    data/dataset.rds")
begin_date <- "2018-01-01"
end_date <- "2020-12-20"
period<-paste(begin_date,"/",end_date,sep="")
stockPrices<-dataset$adjusted[period]
stockPrices <- stockPrices[, 1:9]
## for stocks "AAPL" "ABBV" "AMZN" "DB"   "DIS"  "FB" "GOOG" "HAL"
     "HSBC"
```

2. Retrieve the factor models (3FF):

```
# upload Fama-French factors
```

```
FFdir <- "C:/Users/HP/OneDrive/Escritorio/ANNA/UPC/DSAF/data/"
mydata <- read.csv(paste(FFdir,"F-F_Research_Data_Factors_daily.CSV",sep=""), skip = 4)
mydata <- mydata[-nrow(mydata), ]  # remove last row
fama_lib <- xts(x = mydata[, c(2,3,4)], order.by = as.Date(paste(mydata[, 1]), "%Y%m%d"))
str(fama_lib)
X <- diff(log(stockPrices), na.pad = FALSE)
N <- ncol(X)  # number of stocks
T <- nrow(X)  # number of days

# prepare Fama-French factors
F_FamaFrench <- fama_lib[index(X)]/100
```

3. Construct the Covariance Matrix (Sigma): In the context of the Fama-French 3-factor model, the perturbation matrix Sigma (S) would represent the covariance matrix of the three factors (market, size, and value).

```
# Fama-French 3-factor model
F_ <- cbind(ones = 1, F_FamaFrench)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X))

# Gamma - contiene los coeficientes estimados del modelo de tres factores de Fama-French
colnames(Gamma) <- c("alpha", "beta1", "beta2", "beta3")
alpha <- Gamma[, 1]
B <- Gamma[, 2:4]

# Combine them into a matrix for all factors
FF_returns <- cbind(alpha, B)
# Construct the Sigma Matrix using FF_returns:
Sigma_3FF <- cov(FF_returns)

E <- xts(t(t(X) - Gamma %*% t(F_)), index(X)) # Compute Residuals
PsiFF <- (1/(T-4)) * t(E) %*% E # Compute Residual Covariance
Sigma_FamaFrench <- B %*% cov(F_FamaFrench) %*% t(B) + diag(diag(PsiFF)) # needed for computing the Robust Global Maximum Return Portfolio
```

4. Compute the Robust (Ellipsoid) Global Maximum Return Portfolio. It seeks to maximize the expected return while considering uncertainty or perturbation in the factor model. The ellipsoid framework implies that the portfolio weights are chosen to maximize the expected return within an ellipsoidal uncertainty region defined by the perturbation matrix Sigma (S).

```
mu <- colMeans(X)
portfolioMaxReturnRobustEllipsoid <- function(mu_hat, S, kappa = 0.1) {
  S12 <- chol(S)  # t(S12) %*% S12 = Sigma
  w <- Variable(length(mu_hat))
  prob <- Problem(Maximize( t(w) %*% mu_hat - kappa*p_norm(S12 %*% w,p=2) ),
                  constraints = list(w >= 0, sum(w) == 1))
  result <- solve(prob)
  return(as.vector(result$getValue(w)))
}
```

```
10
11      # A robust solution
12      kappa<-0.63
13      w_GMRP_robust <- portfolioMaxReturnRobustEllipsoid(mu,Sigma_
            FamaFrench,kappa)
14      names(w_GMRP_robust) <- colnames(X)
15      w_all_GMRP_robust_ellipsoid <- cbind(w_GMRP_robust)
```

The implemented solution to obtain the maximum return portfolio considering μ with elliptic constraints given the estimated covariance varies depending on the kappa value.

$$\mu' = \text{colMeans}(X) + k * S^{0.5}u, \quad \|u\| < 1,$$

Kappa values near 0 might be less diversified and more aligned with the traditional view of allocating all the budget on the single stock with largest expected return. On the other hand, a kappa value near 1 emphasizes more diversified but defensively positioned portfolios, achieving robustness against perturbations in the factor model. In order to prove it, the optimization has been done with different kappa values:

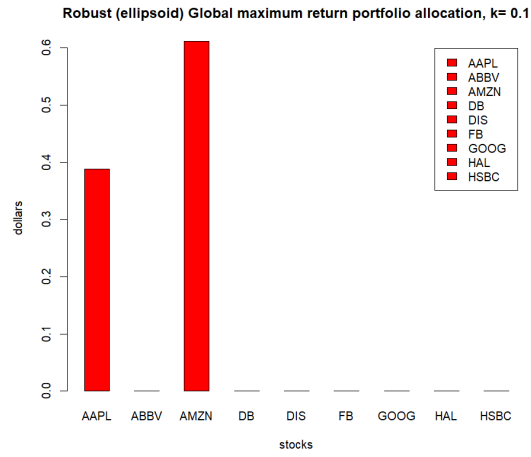
- **kappa=0.1** The optimal solution is to allocate all the budget on AMZN and APPL. Global Maximum Return Portfolio suggests to invest all the budget on a single stock, which goes against diversification. This strategy suggests a concentrated, less diversified approach, focusing on a small number of stocks, potentially exposing the portfolio to higher risk.

- **kappa=0.3** The optimal solution shows more diversification than the previous one. It suggests allocating the budget in different stocks, principally to HSBC and AMZN but diversifying it also in APPL, ABBV, DIS and GOOG. This approach promotes a more diversified portfolio, spreading the budget across a broader set of stocks to mitigate risk.

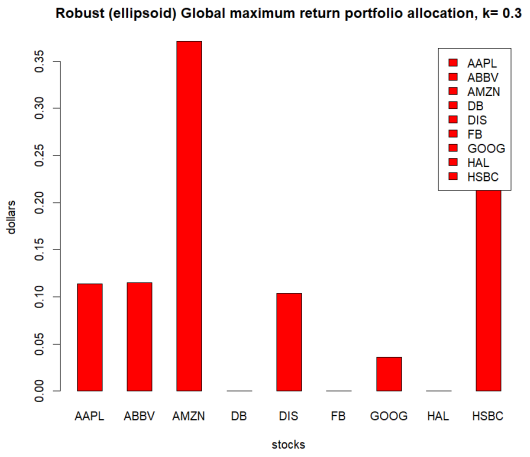
- **kappa=0.63** The idea of diversifying the budget is magnified. The difference between this kappa and the last one is that it shows more confidence in HSBC stock, suggesting the allocation of the budget principally on HSBC and AMZN but still allocating some part of the budget in ABBV, DIS and GOOG. This strategy places a higher emphasis on HSBC, maintaining diversification but expressing increased confidence in a specific stock.

- **kappa=0.9** Emphasizes the diversification of the portfolio, which is defensively positioned to enhance robustness against variations in the factor model.

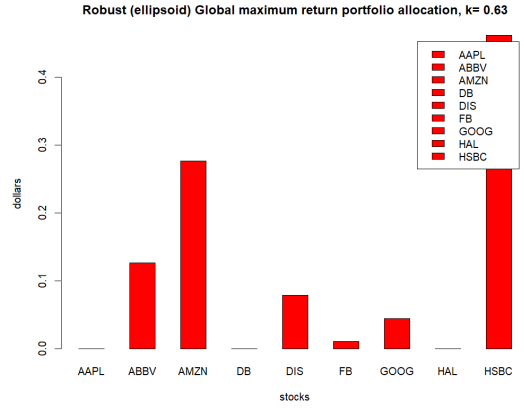
The following figure shows the different Robust Global Maximum Return Portfolio results:



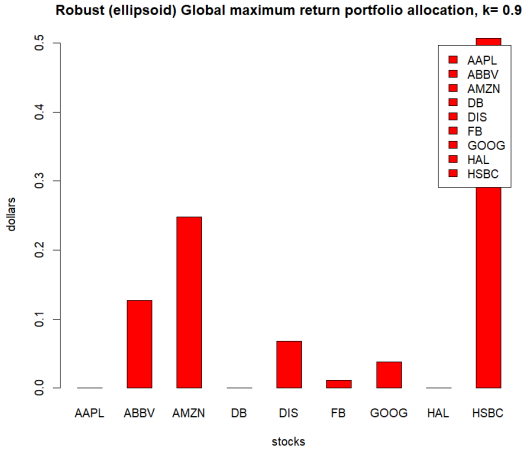
(a) kappa=0.1



(b) kappa=0.3



(c) kappa=0.63



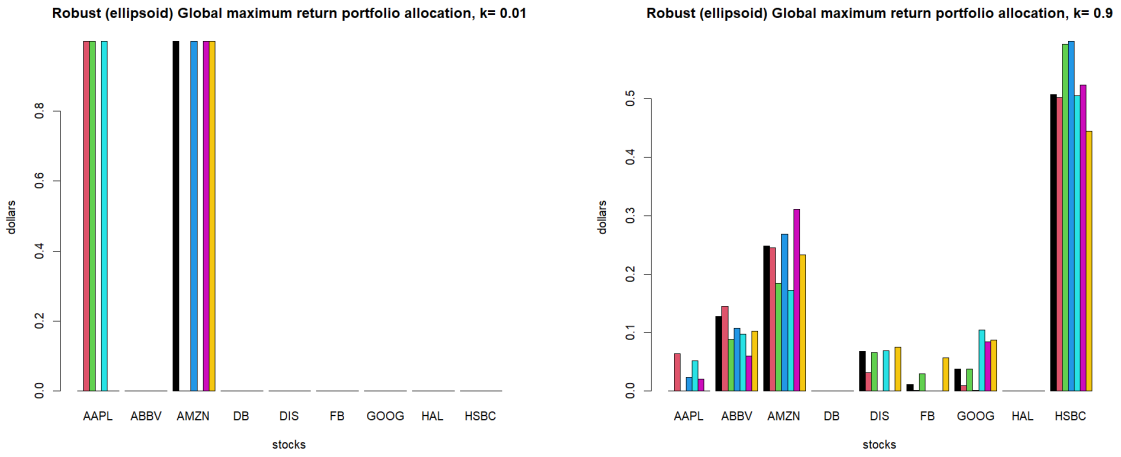
(d) kappa=0.9

Figure 1: Portfolios results depending on kappa value

To check for sensitivity, it has been compute it and then introduced some estimation error in μ to see the effect. If the optimal allocation changes from realization to realization, it means that the obtained portfolio is not useful.

```
# Try multiple robust noisy solutions to check for sensitivity.
set.seed(357)
for (i in 1:6) {
  X_noisy <- rmvnorm(n = T, mean = mu, sigma = Sigma_FamaFrench)
  mu_noisy <- colMeans(X_noisy)
  Sigma_noisy <- cov(X_noisy)

  w_GMRP_robust_ellipsoid_noisy <- portfolioMaxReturnRobustEllipsoid(mu_
      noisy, Sigma_noisy, kappa)
  w_all_GMRP_robust_ellipsoid <- cbind(w_all_GMRP_robust_ellipsoid, w_GMRP_
      robust_ellipsoid_noisy)
}
```

The following plot shows the different portfolios obtained in each realization:

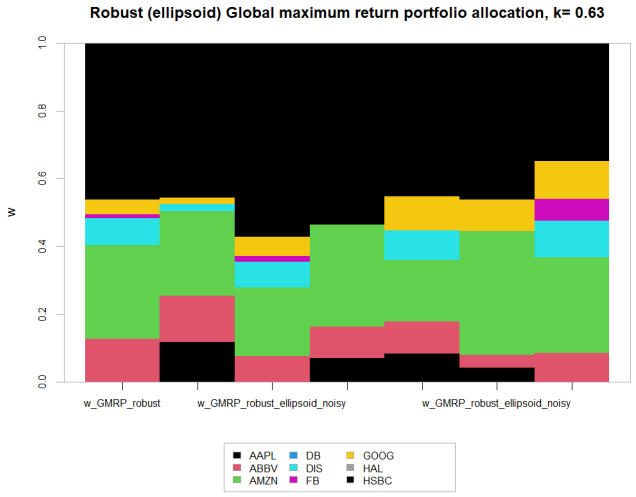


(a) kappa=0.01 (b) kappa=0.9

Figure 2: Sensitivity of RGMR portfolio varying kappa

It is appreciable that the optimal solution behaviour depends on kappa. For a kappa near to 0 (0.01) the optimal allocation changes totally from realization to realization. This scenario is very risky, leading to investing all the budget in one stock or another depending on the realization. On the other hand, for kappa near to 1, the optimal solution suggest a more diversified option which is more desirable. The goal is to enhance robustness against variations, so the kappa 0.9 seems a more stable option.

This plot compares the allocations in the optimal portfolio for kappa=0.63 taking into account different realizations. While the optimal allocation remains subject to variation across different scenarios, the degree of variability is less pronounced compared to cases where kappa approaches 0. In general, it suggest to allocate the budget mainly between HSBC and AMZN. However, there is also a discernible allocation to ABBV, DIS, and GOOG, highlighting a commitment to maintaining a diversified investment strategy.



The next goal of the exercise was to introduce a Sentiment Analysis component to our perturbation matrix. Thus, we have only changed the covariance matrix in our code, the

third point.
3. Construct the Covariance Matrix (Sigma): The perturbation matrix Sigma (S) would represent the covariance matrix of the factor that represents the sentiment on each stock.

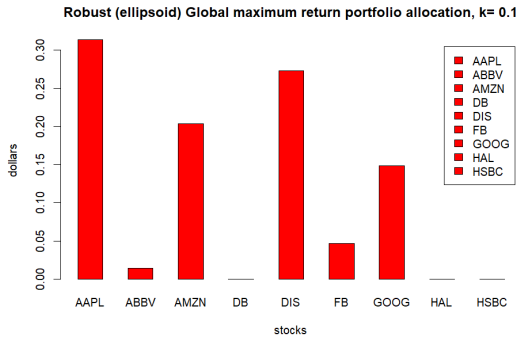
```
##create the PNlog Market index
PNlogMkt <- na.approx(as.xts(rowAvgs(dataset$PNlog),order.by=index(
    dataset$PNlog)))
SentIndx <- PNlogMkt[index(X)]

### Second model
# 1-factor model SentInx
F_ <- cbind(ones = 1, SentIndx)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X))
colnames(Gamma) <- c("alpha", "beta")
alpha <- Gamma[, 1]
beta <- Gamma[, 2]
E <- xts(t(t(X) - Gamma %*% t(F_)), index(X))
Psi_Sent <- (1/(T-2)) * t(E) %*% E
Sigma_SentInx <- as.numeric(var(SentIndx)) * beta %o% beta + diag(diag(
    Psi_Sent))
```

The code used on the following experiments is the same as before, but changing our covariance matrix.

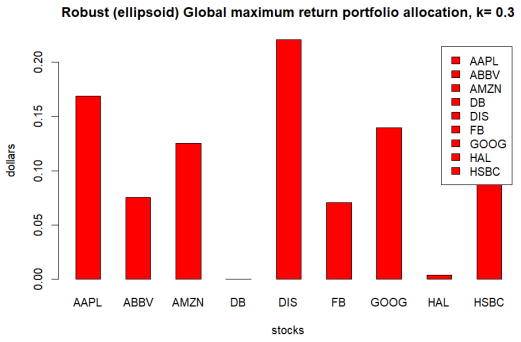
We have also experimented with the value of kappa, and which impact it has on our portfolio. As we mentioned before, a kappa close to 0 will be less diversified, and a kappa close to 1 will have a more defensive portfolio allocation, with robustness against perturbations. We have used the same 4 values of kappa we used with the previous perturbation matrix.

- **kappa=0.1** The optimal solution is to allocate most of the budget on AMZN, APPL, DIS and GOOG. We can see that using a sentiment index as our perturbation leads to a more diversified portfolio even with a small kappa.
- **kappa=0.3** The optimal solution shows more diversification than the previous one. It suggests allocating the budget in different stocks. Still, there are stocks (DIS, APPL, AMZN...) with a higher allocation than others, like DB or HAL.
- **kappa=0.63** Using this kappa has a similar result as the previous kappa, but now, the stocks that have a higher allocation are HBSC and DIS, while others like APPL or AMZN have a lower allocation. Stocks like DB and HAL have now a higher allocation.
- **kappa=0.9** The allocation distribution with this kappa is very similar to our previous one, with our most prominent stocks being HBSC and DIS, but with an approximately even distribution allocation between all stocks

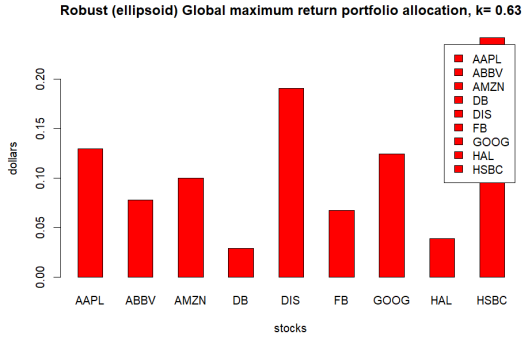
The following figure shows the different Robust Global Maximum Return Portfolio results with the :



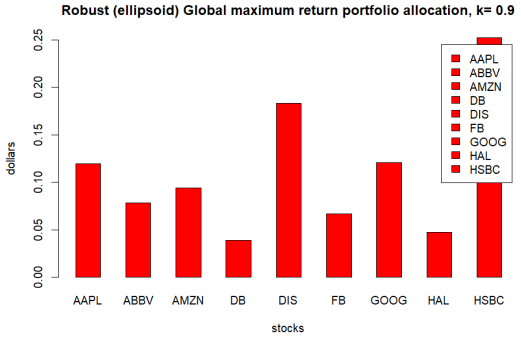
(a) kappa=0.1



(b) kappa=0.3



(c) kappa=0.63



(d) kappa=0.9

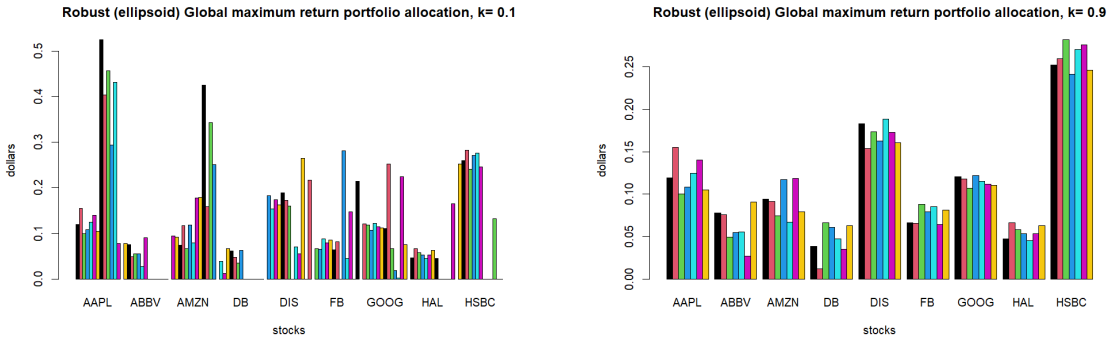
Figure 3: Portfolios results depending on kappa value

To check for sensitivity, it has been compute it and then introduced some estimation error in μ to see the effect. If the optimal allocation changes from realization to realization, it means that the obtained portfolio is not useful.

The following plot shows the different portfolios obtained in each realization:

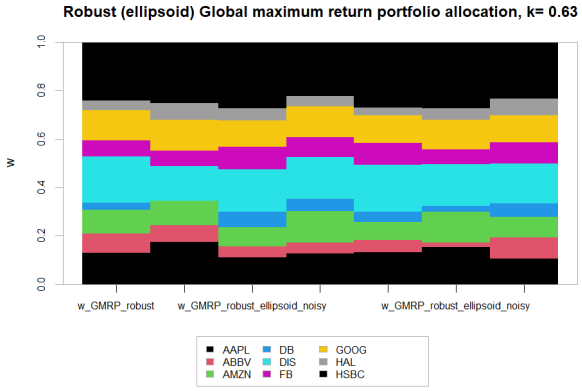
It is appreciable that the optimal solution behaviour depends on kappa. For a kappa near to 0 (0.1) the optimal allocation changes totally from realization to realization. This scenario is very risky, leading to investing all the budget in one stock or another depending on the realization. On the other hand, for kappa near to 1, the optimal solution suggest a more diversified option which is more desirable. The goal is to enhance robustness against variations, so the kappa 0.9 seems a more stable option. We also found this conclusions with the previous perturbation matrix.

This plot compares the allocations in the optimal portfolio for kappa=0.63 taking into account different realizations. While the optimal allocation remains subject to variation across different scenarios, the degree of variability is less pronounced compared to cases where kappa approaches 0. In general, it suggest to allocate the budget between all stocks, with HBSC and DIS having the highest allocation and stocks like GOOG and APPL also have a high allocation. It can be seen that this perturbation matrix leads to a very diversified portfolio.



(a) kappa=0.01 (b) kappa=0.9

Figure 4: Sensitivity of RGMR portfolio varying kappa



After calculating the Robust (ellipsoid) Global maximum return portfolio allocation with two different perturbation matrix, we have seen that they have a high impact on our final portfolio distribution.

Our first covariance matrix portfolio allocation lead to a stock distribution where HBSC's stock has half of our budget, and the other half was diversified between the other stocks. On the other hand, the sentiment index matrix has a more diversified approach. No stock has more than 25% of the budget and we invest some of our budget on all stocks, while the Fama French Matrix did not invest in stocks like AAPL, DB or HAL.