Final Report

-By Team 04

OUTLINE

1. Final choice of submitted strategy

1.1 Introduction of integrated trading system

Our final submitted strategy is a simple trading system consisting of 4 main trading strategies deciding which series we should buy or sell, 2 wager strategies to decide how much we should invest in each series and 5 risk management strategies to prevent extreme risk happening.

We try to harness profit from 4 perspectives; thus 4 individual trading sub-strategies are considered in our final submitted strategy, which are BBands Trend-following Strategy, BBands Mean-reversion Strategy, Triple Moving Average Strategy and Lawrence Macmillan Volatility Trading System. Instructions of which series we should buy or sell are given by them.

1.1.1 Trading Strategy

BBands Trend-following Strategy

In this trend strategy, it is believed that there are small likelihood prices exceed a reasonable range according to statistical ground truth and when it really happens, the herd effect will be followed, thus we should invest in it.

In the code, we long series when price breakthroughs its daily upper bound and short series when price breakthroughs its daily lower bound. We hold when we already have positions and daily prices are still in the trend. Otherwise, we exit.

```r
getOrders.bbtf <- function(store, newRowList, currentPos, info, params){

  # init today's enter positions on each series
  pos <- allzero

  # default exit yesterday's overall positions
  marketOrders <- -currentPos

  # we need to wait until lookback to get the bbands signal
  if (store$iter >= params$lookback){
    for (i in params$series){

      # Calculate everyday bbands upper/lower bounds
      up <- last(BBands(store$cl[1:store$iter,i], sd=params$BBTF_up_sdParam)[,3])
      dn <- last(BBands(store$cl[1:store$iter,i], sd=params$BBTF_dn_sdParam)[,1])

      # enter conditions

      # long enter when cross the upper bound
      if(store$cl[store$iter, i] > up){
        pos[i] <- params$posSizes[i]
      }
      # short enter when cross the lower bound
      else if(store$cl[store$iter, i] < dn){
        pos[i] <- -params$posSizes[i]
      }

      # hold conditions

      if(currentPos[i] > 0){  # when in long position
        # hold when still in the long trend
        if (store$cl[store$iter, i] > store$cl[store$iter-1, i]){
          pos[i] <- currentPos[i]
        }
      }
      else if (currentPos[i] < 0){  # when in short position
        # hold when still in the short trend
        if (store$cl[store$iter, i] < store$cl[store$iter-1, i]){
          pos[i] <- currentPos[i]
        }
      }

      # exit conditions

      if(currentPos[i] != 0 & pos[i] == 0){
        pos[i] <- 0
      }
    }
  }

  marketOrders <- marketOrders + pos

  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=allzero,limitPrices1=allzero,
              limitOrders2=allzero,limitPrices2=allzero))
}
```

BBands Mean-reversion Strategy

The basic trading philosophy of this strategy is similar to ***BBands Trend-following Strategy*** but the price will return to 'normal' range is expected in Mean-reversion strategy.

As can be seen from the code, we short series when price breakthroughs its daily upper bound and long series when price breakthroughs its daily lower bound. We hold when we already have positions and daily prices do not reach the Moving Average target. Otherwise, we exit.

```r
getOrders.bbmr <- function(store, newRowList, currentPos, info, params){
  # init today's enter positions on each series
  pos <- allzero

  # default exit yesterday's overall positions
  marketOrders <- -currentPos

  # we need to wait until lookback to get the bbands signal
  if (store$iter > params$lookback){
    for (i in params$series){

      # calculate everyday bbands upper/lower bounds
      up <- last(BBands(store$cl[1:store$iter,i], sd=params$BBMR_up_sdParam)[,3])
      dn <- last(BBands(store$cl[1:store$iter,i], sd=params$BBMR_dn_sdParam)[,1])

      # everyday re-calculate today's ma30
      sma <- last(SMA(store$op[1:store$iter,i], params$BBMR_SMA))

      # enter conditions

      # short enter when cross the upper bound
      if(store$cl[store$iter, i] > up){
        pos[i] <- -params$posSizes[i]
      }
      # long enter when cross the lower bound
      else if(store$cl[store$iter, i] < dn){
        pos[i] <- params$posSizes[i]
      }

      # hold conditions

      if(currentPos[i] > 0){  # when in long position
        # long hold when not reach MA
        if (store$cl[store$iter, i] > sma){
          pos[i] <- currentPos[i]
        }

      }
      else if (currentPos[i] < 0){  # when in short position
        # short hold when not reach MA
        if (store$cl[store$iter, i] < sma){
          pos[i] <- currentPos[i]
        }
      }

      # exit conditions

      if(currentPos[i] != 0 & pos[i] == 0){
        pos[i] <- 0
      }
    }
  }
  marketOrders <- marketOrders + pos
  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=allzero,limitPrices1=allzero,
              limitOrders2=allzero,limitPrices2=allzero))
}
```

Triple Moving Average Strategy

This trend strategy aims to invest money in clear trends such as the short-term average prices dropping quicker than long-term average prices.

As is shown below, we long series when short-term MA > mid-term MA > long-term MA and short series when short-term MA < mid-term MA < long-term MA. We hold when we already have positions and daily prices do not reach the Moving Average target. Otherwise, we exit.

```r
getOrders.tma<- function(store, newRowList, currentPos, info, params){
  # init today's enter positions on each series
  pos <- allzero

  # default exit yesterday's overall positions
  marketOrders <- -currentPos

  # we need to wait until day 101 so that we have ma100 signal
  if(store$iter > params$lookback){
    for(i in params$series){

      # everyday re-calculate today's
      # short term ma e.g. MA5
      # mid term ma e.g. MA30
      # long term ma e.g. MA100
      sma1 <- last(SMA(store$op[1:store$iter,i], params$TMA_MA1))
      sma2 <- last(SMA(store$op[1:store$iter,i], params$TMA_MA2))
      sma3 <- last(SMA(store$op[1:store$iter,i], params$TMA_MA3))

      # enter conditions

      # long enter when ma5 > ma30 > ma100
      if((sma1 > sma2) &  (sma2 > sma3)){
        # pos[i] <- params$posSizes[i] * 1000000 / store$cl[store$iter,i]
        pos[i] <- params$posSizes[i]
        # short enter when ma5 < ma30 < ma100
      }
      else if((sma1 < sma2) & (sma2 < sma3)){
        pos[i] <- -params$posSizes[i]
      }

      # hold conditions

      # long hold when still in the long trend
      if(currentPos[i] > 0 & store$cl[store$iter, i] > sma1){
        pos[i] <- currentPos[i]
      }
      # short hold when still in the short trend
      else if (currentPos[i] < 0 & store$cl[store$iter, i] < sma1){
        pos[i] <- currentPos[i]
      }

      # exit conditions

      if(currentPos[i] != 0 & pos[i] == 0){
        pos[i] <- 0
      }
    }
  }
  marketOrders <- marketOrders + pos
  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=allzero,limitPrices1=allzero,
              limitOrders2=allzero,limitPrices2=allzero))
}
```

Lawrence Macmillan Volatility Trading System

Volatility is the rate at which stock prices change and can be calculated using the standard deviation formula and by comparing historical volatility over different lengths of time, such as 10, 20 and 50 days, and 100 days. Systematic buying mechanics rules are followed in this strategy:

1. historical volatility is short aligned, i.e., the range of volatility is getting narrower, suggesting the calm before the storm.

2. calculate historical volatility at 5, 10, 20, 30 and 100 days and find its standard deviation.

As demonstrated, we long series when std1 < 0 and std1 < std2 < std3 < std4 < std5 (std1-std5 stands for different lengths of time, from short to long). We hold when we already have positions and daily prices do not reach the Moving Average target. Otherwise, we exit.

```
getOrders.lmv<- function(store, newRowList, currentPos, info, params){
  # init today's enter positions on each series
  pos <- allzero

  # default exit yesterday's overall positions
  marketOrders <- -currentPos

  # we need to wait until day 101 so that we have ema100 signal
  if(store$iter > params$lookback){
    for(i in params$series){

      # everyday re-calculate today's ema at diff. terms
      # e.g. ema5/ema10/ema20/ema30/ema100
      ema1 <- last(EMA(store$cl[1:store$iter,i],params$LMV_EMA1))
      ema2 <- last(EMA(store$cl[1:store$iter,i],params$LMV_EMA2))
      ema3 <- last(EMA(store$cl[1:store$iter,i],params$LMV_EMA3))
      ema4 <- last(EMA(store$cl[1:store$iter,i],params$LMV_EMA4))
      ema5 <- last(EMA(store$cl[1:store$iter,i],params$LMV_EMA5))

      # everyday re-calculate today's std5/std10/std20/std30/std100
      std1 <- store$cl[store$iter,i] - ema1
      std2 <- store$cl[store$iter,i] - ema2
      std3 <- store$cl[store$iter,i] - ema3
      std4 <- store$cl[store$iter,i] - ema4
      std5 <- store$cl[store$iter,i] - ema5

      # enter conditions

      # long enter when the volatility is getting smaller and in the down trend
      if(std1<std2 &
         std2<std3 &
         std3<std4 &
         std4<std5 &
         std1 < 0){
        pos[i] <- params$posSizes[i]
      }

      # hold conditions

      if(pos[i] != 0 | std1 < 0){
        pos[i] <- currentPos[i]
      }

      # exit conditions

      if(currentPos[i] != 0 & pos[i] == 0){
        pos[i] <- 0
      }
    }
  }
  marketOrders <- marketOrders + pos
  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=allzero,limitPrices1=allzero,
              limitOrders2=allzero,limitPrices2=allzero))
}
```

1.1.2 Position sizing (Wager Strategy)

Position sizing, what is also called wager strategy in this report, determine how much we should invest in each series each bet. Therefore, two approaches: Kelly formula and Martingale Strategy are introduced on behalf of long-term and short-term information.

Kelly formula

$$f^* = \frac{bp - q}{b} = \frac{p(b+1) - 1}{b},$$

where:

- f* is the percentage of available funds that should be used for the next bet.
- b is the odds available for the bet (excluding the principal).
- p is the winning rate.
- q is the losing rate, i.e., 1 - p.

As a long-term optimal wager strategy, the Kelly formula is implemented as shown above to calculate how much we should invest in each series in each bet by using our trading system in the long run.

The performance of the final strategy is derived by using the test set (1000 days part1 data + 500 days part2 data) and the following function, and then get the long-term optimal betting ratio as our initial wagers for part 3 data for each of the 10 series in the final submitted strategy.

```r
# **********************************
# performance measurement
perfCalc <- function(test, pfolioPnL){

  lastProfit <- vector(mode="numeric",length=10)
  winTimes <- vector(mode="numeric",length=10)
  WinBalance <- vector(mode="numeric",length=10)
  loseTimes <- vector(mode="numeric",length=10)
  loseBalance <- vector(mode="numeric",length=10)
  tradeTimes <- vector(mode="numeric",length=10)
  gambitRate <- vector(mode="numeric",length=10)
  kelly <- vector(mode="numeric",length=10)
  winRate <- vector(mode="numeric",length=10)
  lossRate <- vector(mode="numeric",length=10)

  pnl <- as.double(pfolioPnL[["pfoliosPnL"]][nrow(pfolioPnL[["pfoliosPnL"]]),2])
  PDratio <- pfolioPnL[["fitAgg"]]
  activeness <- as.double(test[["k"]])

  for (d in 2:1500){
    for (i in 1:10){
      # store money spend for establish position
      if (flagStore[d-1,i] == 1 |
          flagStore[d-1,i] == 2){
        lastProfit[i] <- -moneySpendStore[d, i]
      }
      # when clear position, calculate this trade whether lose or win,
      # when losing, lastProfit = 1 , else 0
      else if(flagStore[d-1,i] == 5){
        if (lastProfit[i] - moneySpendStore[d, i] > 0){
          winTimes[i] <- winTimes[i] + 1
          WinBalance[i] <- WinBalance[i] + (lastProfit[i] - moneySpendStore[d, i])
        }else if (lastProfit[i] - moneySpendStore[d, i] < 0){
          loseTimes[i] <- loseTimes[i] + 1
          loseBalance[i] <- loseBalance[i] + (lastProfit[i] - moneySpendStore[d, i])
        }
        tradeTimes[i] <- tradeTimes[i] + 1
      }
    }
  }

  for (i in 1:10){
    winRate[i] <- winTimes[i]/tradeTimes[i]
    lossRate[i] <- 1-winRate[i]
    gambitRate[i] <- (winTimes[i]/WinBalance[i]) / (loseTimes[i]/loseBalance[i])
    kelly[i] <- ((gambitRate[i]+1)*winRate[i] - 1) / gambitRate[i]
  }

  avg_winRate <- mean(winRate, na.rm = TRUE)
  avg_lossRate <- mean(lossRate, na.rm = TRUE)
  avg_winTimes <- mean(winTimes, na.rm = TRUE)
  avg_loseTimes <- mean(loseTimes, na.rm = TRUE)
  avg_tradeTimes <- mean(tradeTimes, na.rm = TRUE)
  sum_winTimes <- sum(winTimes, na.rm = TRUE)
  sum_loseTimes <- sum(loseTimes, na.rm = TRUE)
  sum_tradeTimes <- sum(tradeTimes, na.rm = TRUE)
  sum_kelly <- sum(kelly, na.rm = TRUE)
  expected_profit <- pnl / sum_tradeTimes

  res <- list('pnl'=pnl,
              'PDratio'=PDratio,
              'activeness'=activeness,
              'winTimes'=winTimes,
              'loseTimes'=loseTimes,
              'tradeTimes'=tradeTimes,
              'WinBalance'=WinBalance,
              'loseBalance'=loseBalance,
              'winRate'=winRate,
              'lossRate'=lossRate,
              'gambitRate'=gambitRate,
              'kelly'=kelly,
              'avg_winRate'=avg_winRate,
              'avg_lossRate'=avg_lossRate,
              'avg_winTimes'=avg_winTimes,
              'avg_loseTimes'=avg_loseTimes,
              'avg_tradeTimes'=avg_tradeTimes,
              'sum_winTimes'=sum_winTimes,
              'sum_loseTimes'=sum_loseTimes,
              'sum_tradeTimes'=sum_tradeTimes,
              'sum_kelly'=sum_kelly,
              'expected_profit'=expected_profit
              )
  return(res)
}
```

Martingale Strategy

Martingale Strategy has been proven profitable at trend trading in short-term. From preliminary analysis, we know that except series 1,6,10 are highly volatile and have no obvious trend in part 1 data, the other series are obviously desirable for trend strategy. Therefore, we double our wager at one series after losing until winning as the code shows.

```r
# **********************************
# Martingale Wager strategy

Martingale <- function(params, store){
  for (i in params$series){
    # store money spend for establish position
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      Martingale_lastProfit[i] <<- -moneySpendStore[store$iter+1, i]
    }
    # when clear position, calculate this trade whether lose or win,
    # when losing, lastProfit = 1 , else 0
    else if(flagStore[store$iter,i] == 5){
      if (Martingale_lastProfit[i] - moneySpendStore[store$iter+1, i] < 0){
        Martingale_lastProfit[i] <<- 1
      }else{
        Martingale_lastProfit[i] <<- 0
      }
    }
    # when clear position and lose, double the wager for the next time
    if (flagStore[store$iter,i] == 5 & Martingale_lastProfit[i] == 1){
      Martingale_multiplier[i] <<- Martingale_multiplier[i] * params$Martingale
    }
    # if win, just reset the wager to the initial amount
    else if (flagStore[store$iter,i] == 5 & Martingale_lastProfit[i] == 0){
      Martingale_multiplier[i] <<- 1
    }
  }
}
```

1.1.3 Risk Management

Risk management is vital to prevent extreme cases happening like anomaly data. Hence, 5 risk management strategies including Stop-gain, Stop-loss and Strategy-exit mechanisms are applied.

Drawdown Stopgain

Drawdown Stopgain is used by comparing the daily drawdown of price and our target to decide whether we should exit our position in one series.

```r
# *********************************
# Drawdown Stop-gain method

# The Drawdown Stop-gain method is a strategy that does not involve
# any Stop-gain strategy in a bull market,
# and then redeems when there is a relatively large short-term Drawdown

Drawdown_Stopgain <- function(params, store, currentPos, Orders){
  for (i in params$series){
    # store money spend at position establish moment
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      Drawdown_lastProfit[i] <<- moneySpendStore[store$iter+1, i]
    }

    else if (flagStore[store$iter,i] == 3
             & store$cl[store$iter, i] > Drawdown_lastProfit[i]/currentPos[i]){
      if ((store$cl[store$iter, i] - store$cl[store$iter-1, i]) /
              store$cl[store$iter-1, i] < params$Stopgain_long_param){
        Orders[i] = -currentPos[i]
      }
    }
    else if(flagStore[store$iter,i] == 4
            & Drawdown_lastProfit[i]/currentPos[i] > store$cl[store$iter, i]){
      if ((store$cl[store$iter, i] - store$cl[store$iter-1, i]) /
              store$cl[store$iter-1, i] > params$Stopgain_short_param){
        Orders[i] = -currentPos[i]
      }
    }
  }
  return(Orders)
}
```

Time Stops

Time stops method is utilized to check if we have been in trade too long in one series by counting days in trade after entering a position.

```r
# *********************************
# Time stops

# # check if we have been in trade too long
#
# # we maintain that pos[i] is an integer
# # if pos[i] == 0 we were flat last period
# # if pos[i] >  0 we have been long  for count[i] periods
# # if pos[i] <  0 we have been short for count[i] periods
#
time_Stops <- function(params, store, currentPos, Orders){
  for (i in params$series){

    # long holding period calculation

    if (flagStore[store$iter, i] == 1) { # in long position today
      timeStops_count[i] <<- 1
    }
    else if (timeStops_count[i] > 0 & Orders[i] == 0){
      timeStops_count[i] <<- timeStops_count[i] + 1
    }

    # short holding period calculation

    if (flagStore[store$iter, i] == 2) { # in short position today
      timeStops_count[i] <<- -1
    }
    else if (timeStops_count[i] < 0 & Orders[i] == 0){
      timeStops_count[i] <<- timeStops_count[i] - 1
    }

    if(timeStops_count[i] == params$long_holdPeriod |
       timeStops_count[i] == -params$short_holdPeriod){  # reached holding period
      Orders[i] <- -currentPos[i] # don't stay short
      timeStops_count[i] <<- 0 # reset count to 0
    }
  }

  return(Orders)
}
```

Trailing Stops

This strategy evaluates if the daily prices decrease over 8% compared with its highest price over holding period, and if we are in a long position, then exit all positions in one series. If the daily prices increased over 8% compared with its lowest price over holding period, and if we are in a short position, then exit all positions in one series.

```r
# ***********************************
# Trailing Stops

# if daily close price decrease over 8% compared with
# Highest Close over holding period,
# and we are in long position, then exit

trailing_Stops <- function(params, store, currentPos, Orders){
  for (i in params$series){
    # store num of days in trade each time
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      trailStops_count[i] <<- 1
    }

    else if(flagStore[store$iter,i] == 3 |
            flagStore[store$iter,i] == 4){
      if (store$cl[store$iter,i] <
            max(store$cl[(store$iter-trailStops_count[i]):(store$iter-1), i]) *
              (1-params$trailingStops_long_param) & currentPos[i] > 0){
        Orders[i] <- -currentPos[i] # don't stay long
      }
      else if (store$cl[store$iter,i] >
                 min(store$cl[(store$iter-trailStops_count[i]):(store$iter-1),
i]) *
                   (1+params$trailingStops_short_param) & currentPos[i] < 0){
        Orders[i] <- -currentPos[i] # don't stay short
      }

      trailStops_count[i] <<- trailStops_count[i] + 1
    }
  }
  return(Orders)
}
```

VaR and ES

The VaR and ES approach to evaluate extreme risk will be tested in Part 3 data each day. If a pre-defined huge loss happened, we exit all positions in one series.

```r
# **********************************
# VaR and ES risk control

# to prevent extreme cases happen
VaR_ES_Stops <- function(params, store, currentPos, Orders){
  for (i in params$series){
    # Calculate returns
    RETS<- diff(log(store$cl[1:store$iter,i]))

    if(VaR(RETS, p=params$VaR_p, method = "gaussian") < params$VaR_ret){
      Orders[i] <- -currentPos[i]
    }
    else if(ES(RETS, p=params$ES_p, method = "gaussian") < params$ES_ret){
      Orders[i] <- -currentPos[i]
    }
  }
  return(Orders)
}
```

Strategy Exit Mechanism

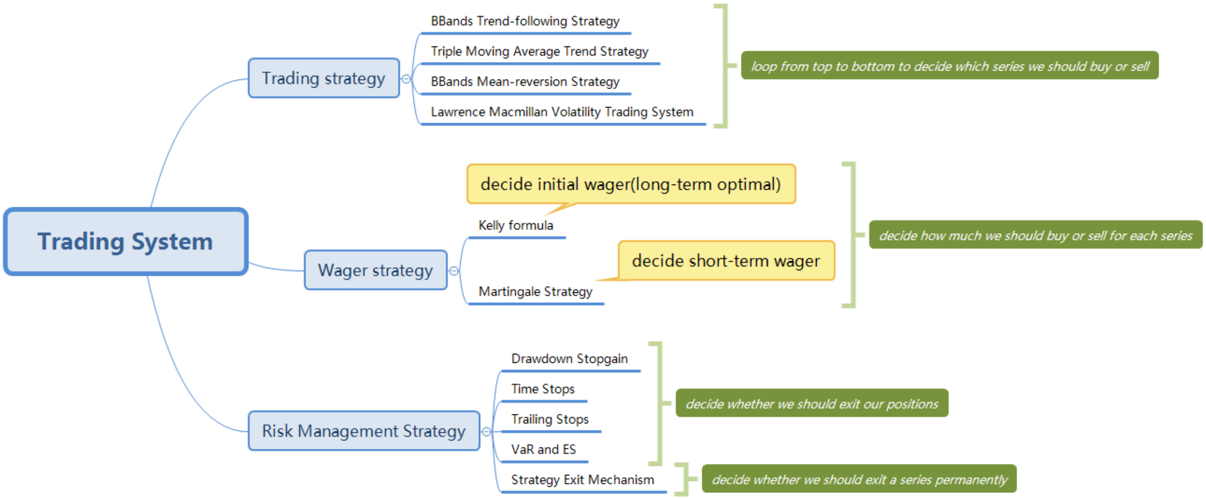
This strategy aims to stop losses when our trading system permanently fails by setting a continuous fail times limit and accumulating money loss target for each series. If one of these two conditions triggered, a series will stop trading forever.

```r
# *********************************
# Strategy Exit Mechanism

Strategy_Exit <- function(params, store){
  for (i in params$series){
    # if strategyExit on series i already triggered then just pass
    if (strategyExit_multiplier[i] == 0){
      next
    }

    # store money spend when enter
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      SE_lastProfit[i] <<- -moneySpendStore[store$iter+1, i]
    }
    # when clear position, calculate this trade whether lose or win,
    # when losing, SE_lastProfit = 1 , else 0
    else if(flagStore[store$iter,i] == 5){
      cum_pnl[i] <<- cum_pnl[i] + (SE_lastProfit[i] - moneySpendStore[store$iter+
1, i])

      if (SE_lastProfit[i] - moneySpendStore[store$iter+1, i] < 0){
        SE_lastProfit[i] <<- 1
      }else{
        SE_lastProfit[i] <<- 0
      }
    }

    if (flagStore[store$iter,i] == 5 & SE_lastProfit[i] == 1){
      lossTimes[i] <<- lossTimes[i] + 1
    }
    # if win, just reset the lossTimes to the initial 0
    else if (flagStore[store$iter,i] == 5 & SE_lastProfit[i] == 0){
      lossTimes[i] <<- 0
    }

    # stop trading particular series
    # if strategy bet wrong params$srategyExit_params times
    if (lossTimes[i] >= params$srategyExit_params){
      strategyExit_multiplier[i] <<- 0
    }

    # stop trading particular series
    # if strategy cumulatively loses more than £50k
    if (cum_pnl[i] < -50000){
      strategyExit_multiplier[i] <<- 0
    }
  }
}
```

1.2 Strategies of different parts working together

Generally, our trading system top-down design is as Graph 1 shows:

Graph 1. Trading System Mechanics




```r
# Combined Strategy  ------------------------

getOrders <- function(store, newRowList, currentPos, info, params){

  # init store

  if (is.null(store)) {   #init store in the first day
    store <- initStore(newRowList,params$series)}
  else{
    store <- updateStore(store, newRowList, params$series)  #after day 1, update everyday data to store
  }

  # init market Orders & limit Orders12 & limit Prices12

  marketOrders <- allzero
  limitOrders1 <- allzero;limitPrices1 <- allzero
  limitOrders2 <- allzero;limitPrices2 <- allzero

  if (store$iter >= params$lookback){

    # A kind Reminder
    # *******************************************
    # store$iter day 0 = moneySpendStore Day 1
    # *******************************************

    # calculate Today's real money spend on each series
    moneySpendCalc(params, store)

    # market condition filter(tbc)
    # 1. if trigger anomoly(e.g. extreme low volume...) then we stop trading for at least 30 days(tbd)
    # 2. ...

    bbtf <- getOrders.bbtf(store, newRowList, currentPos, info, params)
    tma <- getOrders.tma(store, newRowList, currentPos, info, params)
    bbmr <- getOrders.bbmr(store, newRowList, currentPos, info, params)
    lmv <- getOrders.lmv(store, newRowList, currentPos, info, params)


    # decide which strategy should dominate

    for(i in params$series){  # looping series

      # main logic

      if(currentPos[i] == 0 & flagStore[store$iter,i] == 6){
        order_index <- 1
        # looping orders order by Priority
        for(s in orders_4){
          # if orders[s] decides to long or short any shares of stock using market/limit orders
          # then stop and output orders to combined strategy orders
          if(s$marketOrders[i] != 0){
            marketOrders[i] <- s$marketOrders[i]

            # store which strategy is dominated which series everyday, store in codes
            # 1:5 stands for (mm, bbtf, tma, bbmr, lmv) seperately
            SOCStore[store$iter+1,i] <<- order_index
            break
          }
          order_index <- order_index + 1
        }
      }

      else if(currentPos[i] != 0){  # if someone is on charge
        # continue using decisions made by the same strategy
        SOCStore[store$iter+1,i] <<- SOCStore[store$iter,i]
        marketOrders[i] <- orders_4[[SOCStore[store$iter+1,i]]]$marketOrders[i]
      }
    }

    # Wager Strategy

    Martingale(params,store)
    # reversed_Martingale(params,store)

    # Risk Control Strategy

    # Stop-gain Strategy
    marketOrders <- Drawdown_Stopgain(params, store, currentPos, marketOrders)

    # Stop-loss Strategy
    # Time Stops
    marketOrders <- time_Stops(params, store, currentPos, marketOrders)
    # Trailing Stops
    marketOrders <- trailing_Stops(params, store, currentPos, marketOrders)
    # VaR and ES Stops
    marketOrders <- VaR_ES_Stops(params, store, currentPos, marketOrders)

    # Strategy Exit Strategy
    Strategy_Exit(params, store)
  }

  # log everyday on trade signals
  SignalLog(params, store, currentPos, marketOrders, limitOrders1, limitOrder2)

  # update market/limit orders everyday
  marketOrderStore[store$iter+1,] <<- marketOrders  # stores market Orders of strategy on each series everyday

  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=limitOrders1,limitPrices1=limitPrices1,
              limitOrders2=limitOrders2,limitPrices2=limitPrices2))
}
```

Like Graph 1 and the Combined Strategy code shows, every day our trading system runs 4 trading strategies, 5 risk management strategies and 2 wager strategies in sequence. Throughout this daily procedure, the trading decision is derived from daily OHLCV information and trading signals.

The daily OHLCV information is passed by the Backtester every day, updated and stored in sets of intermediate variables. Most trading signals, like Moving Average on our trading strategies, are pre-processed every day using the OHLCV stored in these variables.

```r
# **********************************
# Store OHLCV for feeding backtester

# Store close price
initClStore  <- function(newRowList,series) {
  clStore <- matrix(0,nrow=maxRows,ncol=length(series))
  return(clStore)
}
updateClStore <- function(clStore, newRowList, series, iter) {
  for (i in 1:length(series))
    clStore[iter,i] <- as.numeric(newRowList[[series[i]]]$Close)
  return(clStore)
}

# store open price
initOpStore<- function(newRowList,series){
  opStore<- matrix(0,nrow=maxRows,ncol=length(series))
  return(opStore)
}
updateOpStore<- function(opStore,newRowList,series,iter){
  for(i in 1:length(series))
    opStore[iter,i]<- as.numeric(newRowList[[series[i]]]$Open)
  return(opStore)
}

# store high price
initHiStore<- function(newRowList,series){
  hiStore<- matrix(0,nrow=maxRows,ncol=length(series))
  return(hiStore)
}
updateHiStore<- function(hiStore,newRowList,series,iter){
  for(i in 1:length(series))
    hiStore[iter,i]<- as.numeric(newRowList[[series[i]]]$High)
  return(hiStore)
}

# store low price
initLoStore<- function(newRowList,series){
  loStore<- matrix(0,nrow=maxRows,ncol=length(series))
  return(loStore)
}

updateLoStore<- function(loStore,newRowList,series,iter){
  for(i in 1:length(series))
    loStore[iter,i]<- as.numeric(newRowList[[series[i]]]$Low)
  return(loStore)
}

# store volume
initVolStore<- function(newRowList,series){
  volStore<- matrix(0,nrow=maxRows,ncol=length(series))
  return(volStore)
}

updateVolStore<- function(volStore,newRowList,series,iter){
  for(i in 1:length(series))
    volStore[iter,i]<- as.numeric(newRowList[[series[i]]]$Volume)
  return(volStore)
}

# store full list
initFullstore<- function(newRowList,series){
  fullStore<- vector(mode="list", length=length(series))
  fullStore<- lapply(fullStore, function(x) x=xts(data.frame(Open=0,High=0,Low=0,Close=0,Volume=0), as.Date(0))[-1,])
  return(fullStore)
}

updateFullStore<- function(fullStore,newRowList, series, iter){
  for(i in 1:length(series)){
    NEW<- xts(matrix(as.numeric(newRowList[[series[i]]]),nrow=1),as.Date(index(newRowList[[series[i]]])))
    fullStore[[i]]<- rbind(fullStore[[i]], NEW)
  }
  return(fullStore)
}

# TOTAL DATA STORED IN STORE VARIABLE
initStore <- function(newRowList,series) {
  return(list(iter=0,
              cl=initClStore(newRowList,series),
              op=initOpStore(newRowList,series),
              hi=initHiStore(newRowList,series),
              lo=initLoStore(newRowList,series),
              vol=initVolStore(newRowList,series),
              full= initFullstore(newRowList,series)
              )
        )
}

updateStore <- function(store, newRowList, series) {
  store$iter <- store$iter + 1
  store$cl <- updateClStore(store$cl,newRowList,series,store$iter)
  store$op <- updateOpStore(store$op,newRowList,series,store$iter)
  store$hi <- updateHiStore(store$hi,newRowList,series,store$iter)
  store$lo <- updateLoStore(store$lo,newRowList,series,store$iter)
  store$vol <- updateVolStore(store$vol,newRowList,series,store$iter)
  store$full<- updateFullStore(store$full,newRowList,series,store$iter)
  return(store)
}
```

Trading signals including strategy controlling signals, position signals and real money spent signals.

Strategy controlling signals log which trading strategy is dominating one series at a period of time and daily trading status of one series. These two signals are denoted as SOCStore and flagStore separately in the following code.

```r
# # stores strategy decisions on each series everyday
# 1:long enter, 2:short enter, 3:long hold,
# 4:short hold, 5:clear position, 6:nothing happen
flagStore <- matrix(0,nrow=3100,ncol=10)
# stores strategy on charge on each series, 1:4 for 4 strategies
SOCStore <- matrix(0,nrow=3100,ncol=10)
```

Take SOCStore for instance, on day 221, if only BBands Mean-reversion Strategy triggers and decides to buy 100 shares of series 1, and it exits on day 223. Then values of SOCStore (221, 1), (222,1) and (223,1) would be the sequence encoding of BBands Mean-reversion Strategy, which is 2 showing that from day 221 to 223, BBands Mean-reversion Strategy controls series 1. Therefore, in this period of time, all trading decisions should be made by this strategy in case other strategies increase positions on series 1 or trade on different directions.

Their encoding conditions are shown in the code below:

```r
# *********************************
# Log Trade Signal
# record combined strategy trade signals

SignalLog <- function(params, store, currentPos, marketOrders, limitOrders1, limitOrder2){

  for(i in params$series){

    # enter conditions

    # currentpos is empty
    if(currentPos[i] == 0){
      if(marketOrders[i] > 0){  # if today marketOrder or limitOrder > 0
        flagStore[store$iter+1,i] <<- 1  # long enter flag
      }
      else if(marketOrders[i] < 0){  # if today marketOrder < 0
        flagStore[store$iter+1,i] <<- 2  # short enter flag
      }
    }

    # hold conditions

    # when in long position
    if (marketOrders[i] == 0){
      if(currentPos[i] > 0){
        flagStore[store$iter+1,i] <<- 3  # long hold flag
      }
      # when in short position
      else if (currentPos[i] < 0){
        flagStore[store$iter+1,i] <<- 4  # short hold flag
      }
    }

    # exit conditions

    if (currentPos[i] != 0 & marketOrders[i] != 0){
      flagStore[store$iter+1,i] <<- 5  # exit flag
    }

    # no trading condition

    if (currentPos[i] == 0 & marketOrders[i] == 0){
      flagStore[store$iter+1,i] <<- 6  # no trading flag
      SOCStore[store$iter+1,i] <<- 0
    }
  }
}
```

Similar to Strategy controlling signals, the position signals and real money spent signals, denoted as marketOrderStore and moneySpendStore respectively in the code, are used to store daily market orders and real money spent on market orders on each series. Because we only used market orders to trade in this trading system, these two signals can be considered as the daily orders and money logs of our trading system, which can be utilized to calculate a number of matrixes like Win Rate, Loss Rate, Continuous Loss Times, Win/Loss amount for a single trade. These matrices play an important role in our Wager Strategy and Risk Management Strategy.

```r
# Helper global variables------------

# for init vector

allzero  <- rep(0,10) # used for initializing vectors
maxRows <- 3100  # set maxRows as the number of rows in data

# # stores strategy decisions on each series everyday
# 1:long enter, 2:short enter, 3:long hold,
# 4:short hold, 5:clear position, 6:nothing happen
flagStore <- matrix(0,nrow=3100,ncol=10)
# stores strategy on charge on each series, 1:4 for 4 strategies
SOCStore <- matrix(0,nrow=3100,ncol=10)

# store everyday orders and real money spent for each series

moneySpendStore <- matrix(0,nrow=3100,ncol=10) # stores real money spent of strategy on each series everyday
marketOrderStore <- matrix(0,nrow=3100,ncol=10) # stores market orders on each series everyday

# some global variables for wager strategy and risk strategies

Martingale_multiplier <- rep(1,10) # Martingale multiplier
Martingale_lastProfit <- vector(mode="numeric",length=10) # stores profit in the last trade for each series over time
revMartingale_lastProfit <- vector(mode="numeric",length=10) # stores profit in the last trade for each series over time
timeStops_count <- rep(0,10) # for Time Stops holding period counting
trailStops_count <- vector(mode="numeric",length=10) # stores number of days in trade
SE_lastProfit <- vector(mode="numeric",length=10) # stores profit in the last trade for each series over time for Strategy Exit
lastProfit <- vector(mode="numeric",length=10) # stores profit in the last trade for each series over time
lossTimes <- vector(mode="numeric",length=10) # stores loss times
strategyExit_multiplier <- rep(1,10) # Strategy Exit multiplier
Drawdown_lastProfit <- vector(mode="numeric",length=10) # stores profit in the last trade for each series over time for Drawdown Stop Gain
cum_pnl <- vector(mode="numeric",length=10)  # stores cumulative pnl
```

Take Drawdown Stop-gain method code as an example to demonstrate how these signals contribute to Risk Management Strategy.

Like the following code indicates, we store total money spent on series 'i' when entering a position as 'Enter Cost' where flagStore is 1 or 2(stands for long/short enter) with our variable 'moneySpendStore' at day 'store$iter+1' on 'Drawdown_lastProfit[i]'. When trading strategy decides to hold our positions on series i(flagStore = 3 or 4, which stands for long/short hold), this risk strategy needs to calculate the rolling gain drawdown to decide if we should exit instantly.

```r
# **********************************
# Drawdown Stop-gain method

# The Drawdown Stop-gain method is a strategy that does not
# involve any Stop-gain strategy in a bull market,
# and then redeems when there is a relatively large short-term Drawdown

Drawdown_Stopgain <- function(params, store, currentPos, Orders){
  for (i in params$series){
    # store money spend when enter position
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      Drawdown_lastProfit[i] <<- moneySpendStore[store$iter+1, i]
    }

    else if (flagStore[store$iter,i] == 3
             & store$cl[store$iter, i] > Drawdown_lastProfit[i]/currentPos[i]){
      if ((store$cl[store$iter, i] - store$cl[store$iter-1, i]) /
          store$cl[store$iter-1, i] < params$Stopgain_long_param){
        Orders[i] = -currentPos[i]
      }
    }
    else if(flagStore[store$iter,i] == 4
            & Drawdown_lastProfit[i]/currentPos[i] > store$cl[store$iter, i]){
      if ((store$cl[store$iter, i] - store$cl[store$iter-1, i]) /
          store$cl[store$iter-1, i] > params$Stopgain_short_param){
        Orders[i] = -currentPos[i]
      }
    }
  }
  return(Orders)
}
```

1.3 Optimizing and checking the robustness of strategy

Except for the 'lookback' and 'series' in the last parameter set we define directly as '100' and '1:10', because of the longest-term MA signal limit in the LMV strategy. If 'lookback' is below 100 days, we cannot get MA100, thus we do not make any transactions in the first 100 days. In addition, 4 trading strategies for 10 series in the training set and test set data have different performance, and if the series are filtered based on different winning strategies, it will lead to high development costs, so we do not put any restrictions on 'series'.

Regarding optimization method, genetic algorithm and 'Multi-mcga' R package are implemented by randomly generating 35 parameters in pre-defined times of 'evolution' and finally selecting the best-K performance parameter sets as the code below indicates.

```r

epoch <- 0

evalFunc <- function(x) {
  s=Sys.time()  #timer
  source('D:/preloading.R');

  epoch <<- epoch + 1

  params.opt <- list(lookback=100,  # watch window, default 100, means no trade at the first 100 days
                     series=1:10,  # contol which series we want to trade, default 1:10
                     posSizes=c(x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8],x[9],x[10]),  # control series viable to TMA
                     long_holdPeriod=x[11],  # control maximum long holding period
                     short_holdPeriod=x[12],  # control maximum short holding period
                     TMA_MA1=x[13], TMA_MA2=x[14], TMA_MA3=x[15],  # control short/Mid/Long term signals for TMA
                     LMV_EMA1=x[16], LMV_EMA2=x[17], LMV_EMA3=x[18], LMV_EMA4=x[19], LMV_EMA5=x[20],  # control short/Mid/Long term signals for TMA
                     BBMR_SMA=x[21],  # control exit SMA for BBMR
                     BBMR_up_sdParam=x[22], BBMR_dn_sdParam=x[23],  # control Upper/Lower bounds for BBMR
                     BBTF_up_sdParam=x[24], BBTF_dn_sdParam=x[25],  # control Upper/Lower bounds for BBTF
                     srategyExit_params=x[26],  # controls maximum acceptable failing times for combined strategy
                     VaR_p=x[27], VaR_ret=x[28],  # control extreme risk measurement for VaR
                     ES_p=x[29], ES_ret=x[30],  # control extreme risk measurement for ES
                     trailingStops_long_param=x[31],  # control trailing history bound when Long Hold
                     trailingStops_short_param=x[32],  # control trailing history bound when Short Hold
                     Stopgain_long_param=x[33],  # control short-term Drawdown Level when Long Hold
                     Stopgain_short_param=x[34],  # control short-term Drawdown Level when Short Hold
                     Martingale=x[35])  # control martingale multiplier

  test.opt <- backtest(dataList, getOrders, params.opt, sMult=0.2)
  pfolioPnL.opt <- plotResults(dataList, test.opt, plotType='ggplot2',
                               titleString=paste('Strategy Performance, Epoch:',epoch))

  final_perf <- perfCalc(test=test.opt, pfolioPnL=pfolioPnL.opt)

  result <- numeric(4)
  result[1] <- -final_perf$pnl
  result[2] <- -final_perf$PDratio
  result[3] <- -final_perf$activeness
  result[4] <- -final_perf$expected_profit

  cat('****************************************')
  cat('\n')
  cat('epoch:',epoch)
  cat('\n')
  e=Sys.time()
  cat('runtime: ', e-s)
  cat('\n\n')
  cat('params:')
  print(params.opt)
  cat('pnl:', -result[1])
  cat('\n')
  cat('PDRatio:', -result[2])
  cat('\n')
  cat('avtiveness:', -result[3])
  cat('\n')
  cat('expected_profit:', -result[4])
  cat('\n')
  cat('****************************************')
  cat('\n')

  return(result)
}

# constrainFunc <- function(x) {
#   x[11] >= 1
#   x[11] <= 3
#   x[12] <= 8
#   x[12] >= 1
# }

res <- nsga2(fn = evalFunc, idim = 35, odim = 4, cdim=0,   # constraints=constrainFunc
             popsize = 100, generations=3,
             lower.bounds=c(0,0,0,0,0,0,0,0,0,0,  # posSizes
                            1,1,2,20,50,2,10,30,60,80,  # params 11-20
                            2,1,1,1,1,4,0.8,-0.2,0.8,-0.2,  # params 21-30
                            0.01,0.01,-0.2,0.01, 1.1),  # params 31-34
             upper.bounds=c(4000,200,100000,2000,1000,18000,1000,1200,1200,240,
                            10,10,10,50,100,10,30,60,80,100,
                            50,5,5,5,5,12,0.999,-0.01,0.999,-0.01,
                            0.2,0.2,-0.01,0.2, 3))

#
# Result <- cbind(res[['par']],res[['value']])
# Result <- cbind(Result, res[['pareto.optimal']])
# write.csv(Result, "D:/result.xls")

# x <- unlist(Result[1,1:40])
```

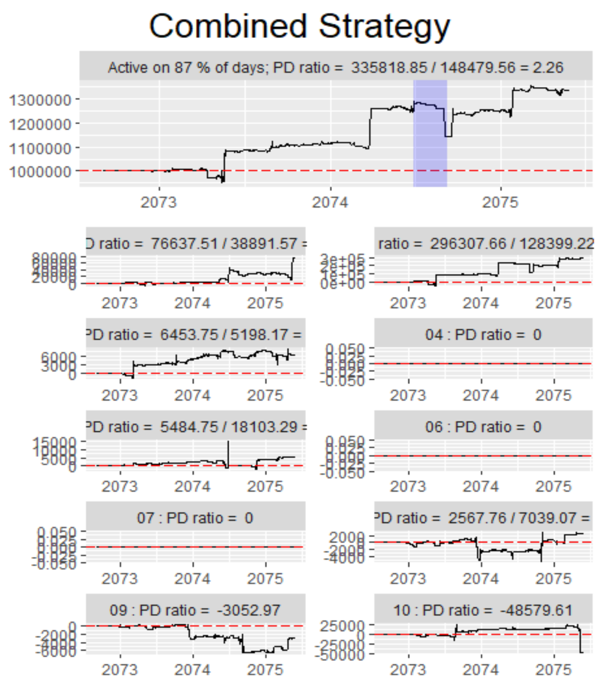
In this process, the most important thing is to define the objective function. At first, we defined the objective function as maximizing PNL and PD ratio, and among the final 100 sets of parameters produced after 100 iterations, we found that these parameter combinations all clearly overfitting, for example, most of them having significantly high profits in the training set (part 1 data), but the profits and positions as well as the number of trades were concentrated in minor series. What's worse, most of the parameter portfolios are inactive and stop trading after making some money. Not surprisingly, in the test set (part 2 data) performance, the 100 final results all showed completely different and relatively large drawdowns and losses, and were very volatile over the trading period. Next, we adjusted the objective function and added two new indicators, Activeness and Expect Profit into the objective function, in order to make our parameter combinations trade more frequently, rather than relying on minor times of trading to make money and then stop trading afterwards to ensure that the final parameter combinations are robust enough, rather than random luck. By bringing in Expect Profit, the maximization of expected profit in each trade can also guarantee our long-term performance is robust. In the meantime, we adjusted the train set to 1000 days data of part 1 plus 500 days data of part 2 and the population sizes of Genetic Algorithm to 100 per generation and 100 generations in total to ensure there are enough mutations of parameters and the information of 10 series during part 2 data.

Table 1. Final Top-100 Performance Parameter Sets after Genetic Algorithm optimization

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	V29	V30	V31	V32	V33	V34	V35	V36	V37	V38	V39	V40
1	2072.618	102.5478	45666.71	792.3253	776.9695	11747.4	145.7095	947.4759	259.774	48.48732	3.570596	7.728174	5.013293	20.92788	85.0485	8.261037	19.14279	41.1023	72.24701	97.1579	9.700385	3.897917	3.17256	3.56702	4.933159	9.005184	0.871702	-0.09575	0.943901	-0.02801	0.10182	0.12245	-0.19453	0.035237	2.98558	-229515	-4.33	-94	-13500.9	1
2	1676.961	63.13995	69389.7	1161.446	837.9512	16641.51	550.4345	75.68437	551.1331	125.6317	4.441696	3.641709	7.959136	23.96201	57.0327	4.373955	21.13592	55.26775	72.38217	98.90077	22.97847	3.740805	4.531964	3.588115	2.874036	10.9402	0.806655	-0.17637	0.818993	-0.01111	0.091461	0.11124	-0.05467	0.126156	2.663517	-1149222	-3.55	-94	-164175	1
3	723.9151	9.987807	71058.2	1840.317	268.1471	12726.92	559.4596	305.6596	839.153	46.68702	7.966441	4.732427	6.721182	25.00363	83.45687	7.200347	26.38177	40.1525	74.75545	95.40991	9.49582	3.690365	2.697836	1.983312	4.841688	7.161683	0.864708	-0.09763	0.927815	-0.04703	0.193057	0.010507	-0.02671	0.027459	2.062164	-1739378	-5.62	-94	-3464.9	1
4	3818.925	106.3874	20168.68	1594.009	793.061	2565.066	717.77	666.397	272.0966	147.8195	8.407549	6.760296	9.656808	34.25904	59.53921	3.145358	13.34837	49.34207	67.83305	88.6366	11.26729	1.356065	4.039993	3.082121	1.714325	7.661991	0.802769	-0.10222	0.900756	-0.16573	0.192595	0.087477	-0.0388	0.151275	2.936747	-1.10E+07	-2.44	-94	-19062.9	1
5	3027.507	173.0519	57931.32	749.3678	479.7175	5355.24	24.54012	340.8206	364.01	93.3124	5.791573	7.797669	7.908665	24.57124	98.893	6.537638	17.59815	38.5969	75.45864	95.49905	26.91509	4.283283	3.625357	4.675256	2.778	6.558415	0.843134	-0.13228	0.811457	-0.03672	0.087975	0.189036	-0.07053	0.081601	2.88397	-6117306	-0.8	-94	-86159.2	1
6	1508.46	3.406986	69389.7	1161.446	837.9512	14406.04	550.4345	75.68437	822.6161	128.3185	4.751156	3.454502	6.763727	22.79195	57.11836	4.373955	21.16601	55.06935	72.38217	98.90077	22.47092	3.944695	3.853309	3.327073	3.204856	7.048522	0.806655	-0.17637	0.818993	-0.01111	0.091461	0.11124	-0.05574	0.135158	2.663517	-1141302	-3.62	-94	-163043	1
7	1676.961	60.60197	69389.7	1613.816	931.7372	16641.51	544.6672	75.68437	542.6942	147.8423	4.441696	3.650952	9.016285	23.96506	57.0327	4.558272	21.27135	50.05329	72.38217	88.63783	18.30187	3.740805	1.193602	4.074186	2.826498	10.9402	0.806655	-0.17895	0.818993	-0.02871	0.092879	0.109221	-0.05386	0.126417	2.663517	-7000253	-1.78	-94	-45162.9	1
8	271.5933	19.15446	43024.09	1550.67	460.6351	13619.49	184.7956	242.4121	1117.02	139.1881	1.073974	2.01997	6.629404	42.58466	54.29554	6.931697	25.95331	48.98931	64.54847	92.52679	26.64733	2.769504	2.644121	3.287291	2.855044	11.78177	0.986489	-0.16649	0.847117	-0.12307	0.198551	0.07018	-0.02256	0.197422	2.432643	-2989586	-3.92	-94	-17181.5	1
9	57.61408	14.72069	48182.9	7.095955	660.2426	14026.6	236.6215	89.13179	481.2501	56.47479	3.56292	1.035746	5.014526	22.42723	78.70059	7.755615	20.84297	41.1023	67.50149	87.32067	12.58401	4.339132	2.840188	4.67822	4.498575	6.060038	0.870467	-0.15125	0.943901	-0.01019	0.095306	0.118235	-0.19137	0.179926	2.97612	-217795	-3.11	-94	-21779.5	0
10	56.29732	156.5778	45875.65	1003.055	751.1609	781.3546	877.1741	213.2456	483.3329	235.1824	6.217549	1.146827	2.278558	24.7875	79.77752	6.343227	12.59851	45.11981	67.24037	86.59237	16.59701	3.872906	3.29517	4.436022	4.858326	11.60424	0.865242	-0.15648	0.893245	-0.13504	0.083483	0.014338	-0.06854	0.068628	1.987643	-7982650	-2.08	-94	-14178.8	0
11	947.8275	15.69599	44683.2	1915.014	498.6721	14818.6	671.6974	190.6715	773.2157	87.65952	1.033688	1.085401	2.753263	38.03915	90.54509	6.869011	19.34433	49.27499	63.70884	85.28602	27.6059	4.692018	2.599645	2.547558	2.534001	8.759155	0.876421	-0.18513	0.944584	-0.02138	0.018402	0.128907	-0.04031	0.156775	2.299896	-1121899	-1.28	-80	-15160.8	0
12	1809.972	67.17763	71513.08	1161.446	837.9512	16641.51	550.4345	74.38147	551.1331	129.9146	4.475695	1.010402	7.137634	23.96201	83.8212	4.373955	21.13592	55.24122	71.71072	98.90077	22.97847	3.740805	4.51083	3.776099	2.874036	10.90353	0.814572	-0.17644	0.884889	-0.02115	0.148035	0.025481	-0.06875	0.126156	2.663517	-796842	-3.35	-81	-18972.4	0
13	3548.032	92.03763	49037.4	859.7043	769.632	3267.256	860.8709	108.5126	146.0419	180.094	3.130479	8.58722	7.745017	20.94493	74.46217	5.835251	26.38329	52.25149	66.51733	93.86273	45.08866	4.105512	1.947066	2.612691	2.699555	5.186916	0.953414	-0.01752	0.927253	-0.07314	0.065843	0.149367	-0.08406	0.119593	2.813461	-127664	-2.33	-94	-31916	0
14	1065.714	92.56158	23032.31	521.194	111.5896	4199.138	49.27059	318.4929	627.2268	83.02331	9.033327	8.597068	8.663627	49.23992	94.10383	5.434701	25.38786	50.25911	65.98134	85.58844	27.04911	2.345376	1.976653	4.147695	4.63601	7.87167	0.937506	-0.12736	0.959036	-0.19997	0.068847	0.060091	-0.08802	0.179725	2.727632	-1968601	-2.64	-94	-6628.29	0
15	212.0931	178.4065	57410.99	1569.357	633.5448	1334.149	271.3318	39.84287	635.3385	109.9672	4.114505	1.146827	8.046911	44.25252	89.62681	6.019174	28.02704	58.31952	65.10063	91.29174	6.755391	4.454128	1.769728	2.99156	4.793435	11.02466	0.853785	-0.16461	0.884273	-0.16707	0.105378	0.010406	-0.16479	0.068628	1.904503	-1235236	-3.44	-94	-4508.16	0
16	3913.698	77.81513	34006.58	1268.568	271.7517	2144.55	458.475	998.9934	1080.998	232.6835	8.290586	7.727157	9.406769	25.54042	85.39167	2.908156	24.62909	41.88325	68.2062	99.0909	23.79209	2.662124	3.115594	3.009844	2.985322	9.424388	0.953685	-0.18873	0.800616	-0.03627	0.077405	0.117557	-0.1918	0.128492	2.936966	-4748215	-1.05	-47	-35700.9	0
17	2357.339	145.9292	74420.32	1376.65	326.765	16645.59	313.3066	805.053	1093.284	86.59239	7.798126	3.927248	9.703346	41.38775	81.792	5.65982	25.49439	41.32649	67.86807	91.4348	15.59094	3.14641	1.594498	4.744034	4.185201	5.357976	0.967843	-0.02357	0.998356	-0.07626	0.056188	0.058821	-0.10904	0.14301	2.881619	-684183	-2.55	-94	-16290.1	0
18	12.25042	85.01606	71671.83	1832.084	268.1471	9098.483	559.4596	1051.412	946.5712	40.34156	7.849895	4.592692	6.993187	25.00363	81.25473	4.6592	25.69475	58.32745	72.54121	93.86413	11.25102	3.983025	1.957429	2.24811	2.745044	6.423657	0.947408	-0.10411	0.927815	-0.02677	0.069829	0.175816	-0.1414	0.052414	2.027282	-508969	-3.64	-94	-6696.96	0
19	57.61408	14.72069	48182.9	7.095955	762.7838	15077.59	253.1614	89.13179	481.2501	56.47479	3.56292	1.035746	5.014526	22.42723	78.70059	7.755615	20.84297	41.1023	67.50149	87.32067	16.18201	3.897917	2.840188	4.67822	4.498575	6.060038	0.870467	-0.15877	0.943901	-0.01019	0.095306	0.118235	-0.19137	0.185272	2.964268	-499447	-1.45	-89	-23783.2	0
20	2668.099	102.5478	46202.25	785.4904	767.5952	17976.56	408.8149	923.3041	268.9977	32.4152	3.56292	1.001925	5.014526	20.93675	74.90223	7.755615	19.85846	41.1023	72.24701	85.81329	9.700385	3.739409	3.17256	4.551415	4.933159	8.712872	0.882507	-0.10543	0.943901	-0.02423	0.109735	0.115977	-0.19834	0.035237	2.98558	-152249	-1.62	-94	-15224.9	0
21	3986.037	114.2869	44135.15	1899.091	616.3196	1229.837	753.702	994.3681	959.3772	221.8573	6.178319	6.228239	2.006331	28.36177	73.21218	6.519776	11.43013	49.31485	66.75281	87.14313	13.8144	1.688769	2.637264	2.33784	2.494031	8.326086	0.801821	-0.11665	0.90592	-0.15659	0.103664	0.085367	-0.03204	0.151635	2.006681	-1081432	-2.04	-94	-2059.87	0
22	1629.307	1.109168	36195.32	1559.359	345.0612	15905.9	552.074	79.04372	823.9734	54.70876	1.357285	3.810989	9.181888	22.68229	64.86006	7.669843	24.4136	34.55299	71.99263	97.45774	22.86469	3.826771	1.240622	1.900957	1.559619	5.795006	0.845935	-0.0556	0.87963	-0.10323	0.13573	0.165976	-0.08885	0.020922	2.132716	-604096	-1.71	-94	-2696.86	0
23	212.0931	77.2734	44376.9	1131.742	633.5448	1334.149	270.6216	39.84287	595.2927	222.4241	9.326491	1.146827	2.534621	31.12582	89.62681	5.401451	22.65847	58.31952	67.65503	85.86111	7.441381	2.916114	3.079885	3.98295	4.794303	11.86219	0.852334	-0.16392	0.909845	-0.0411	0.144761	0.019887	-0.1661	0.068628	1.830639	-515547	-2.43	-94	-1636.66	0
24	2072.618	102.5478	48671.89	785.4904	776.9695	17976.56	408.8149	947.4759	268.9977	36.14241	3.570596	1.00554	5.014526	20.92788	80.17249	7.755615	19.85846	41.1023	72.24701	87.01124	9.700385	3.897917	3.17256	4.511355	4.933159	9.005184	0.882507	-0.09575	0.943901	-0.02801	0.10182	0.129304	-0.19397	0.035237	2.98558	-125185	-1.91	-94	-7363.84	0
25	3913.698	142.5077	34006.58	1178.958	271.7517	15799.04	598.3712	998.9934	1080.998	232.6835	9.045093	7.727157	2.184926	25.77135	85.39167	2.790642	23.04084	42.29411	67.61588	82.66209	22.79612	2.839715	2.328194	3.989263	2.985322	9.464103	0.953685	-0.18792	0.817786	-0.03627	0.077405	0.117557	-0.1918	0.128492	2.973407	-3109608	-0.56	-42	-32391.7	0
26	1028.269	106.3874	20168.68	1594.009	793.061	669.5692	765.3092	1151.344	269.8709	147.8195	8.407549	5.890095	9.656808	34.25904	80.81365	2.762157	20.49057	49.34207	67.99284	88.76886	11.26729	1.454099	2.504907	3.10541	1.756012	6.362623	0.822547	-0.10222	0.900756	-0.19948	0.190051	0.057247	-0.08861	0.141034	2.047988	-756679	-2.27	-94	-1477.89	0
27	3548.032	92.03763	49037.4	859.7043	644.7782	3267.256	860.8709	108.5126	146.0419	180.094	3.130479	8.58722	7.745017	20.94493	74.46217	5.835251	26.38329	52.25149	66.51733	93.86273	40.50245	4.105512	1.947066	2.612691	2.699555	5.186916	0.953414	-0.01752	0.927253	-0.07314	0.065843	0.149367	-0.08406	0.119593	2.813461	-104505	-2.24	-94	-26126.1	0
28	1044.478	81.57444	17468.32	1876.993	109.3444	4111.553	919.9329	323.0552	520.1404	78.93014	9.008881	8.357373	8.859216	44.58144	95.78441	8.063697	17.41236	50.97675	65.60307	85.43618	23.30706	1.645043	2.050379	4.054657	4.794176	7.768666	0.938002	-0.10703	0.931512	-0.19997	0.113747	0.040835	-0.03567	0.179725	2.732296	-2397564	-1.62	-94	-5833.49	0
29	3818.925	106.3874	20168.68	1594.009	793.061	2565.066	717.77	746.5455	272.0966	147.8195	8.407549	6.760296	9.656808	34.25904	59.53921	3.145358	13.34837	49.19933	67.83305	86.47445	11.26729	1.356065	4.039993	3.082121	1.587523	7.661991	0.802769	-0.10222	0.900756	-0.16573	0.192595	0.087477	-0.0507	0.130054	2.936747	-7698055	-1.16	-94	-13295.4	0
30	1044.478	81.57444	23032.31	1883.286	111.5896	4199.138	64.12369	318.4929	612.4062	78.93014	9.033327	8.357373	8.760229	49.23992	95.78441	8.467634	19.00334	50.91624	62.33869	85.58844	25.1817	2.345376	1.976653	4.147695	4.695804	6.789762	0.938002	-0.13352	0.959547	-0.19997	0.11418	0.060091	-0.04917	0.179725	2.727632	-3736857	-1.24	-94	-13840.2	0
31	3955.561	137.5202	72282.54	1448.671	326.765	16641.33	13.23386	805.053	1116.195	72.95958	7.71694	4.020672	9.535722	24.96859	73.63751	5.639839	11.05527	41.32649	66.5784	98.54349	22.05829	3.717012	1.594498	4.860538	2.787943	5.357976	0.896076	-0.12951	0.997977	-0.01122	0.032575	0.05471	-0.11116	0.176412	1.207141	-6754.83	-2.82	-94	-6754.83	0
32	1120.486	91.82741	23032.31	946.345	109.3444	4111.553	899.0107	325.4576	607.4948	80.66173	9.008881	8.357373	8.859216	44.57032	95.78441	8.467634	17.41236	53.97638	62.33869	84.82031	23.30706	1.645043	1.464632	2.51304	4.794176	6.789762	0.936667	-0.11517	0.931512	-0.19998	0.199695	0.193266	-0.02926	0.174185	2.693623	-688343	-1.41	-94	-2742.4	0
33	212.0931	115.8598	44407.73	1419.361	633.5448	1334.149	262.7562	39.84287	635.3385	223.3579	4.033806	1.657582	8.44543	31.12582	89.62681	5.519016	12.59851	58.37178	65.18478	85.51928	7.441381	2.792231	3.079885	2.99156	4.708569	11.99803	0.852817	-0.16392	0.908828	-0.14254	0.122054	0.019758	-0.16478	0.068628	1.840264	-515692	-1.95	-94	-1530.24	0
34	461.9973	97.78559	72122.17	1441.605	270.3981	12199.97	74.01868	870.7567	157.1764	102.2763	3.652582	8.749002	7.981965	38.55095	50.25621	2.885156	27.52888	34.03499	67.67678	85.43296	5.77579	1.763529	1.14772	3.5156	2.139423	4.59815	0.813435	-0.03858	0.968479	-0.16984	0.195989	0.140994	-0.15057	0.095384	2.949649	-694775	-1.23	-94	-11028.2	0
35	2702.137	1.883359	59211.48	81.31517	787.4843	3928.514	253.1614	36.29954	281.166	21.2267	2.932087	6.496468	5.014526	20.93675	78.70059	7.17173	14.83824	41.1023	67.49765	88.46624	9.700385	4.247627	3.089635	4.080146	4.933159	6.977513	0.882744	-0.09399	0.900626	-0.02283	0.095306	0.065791	-0.17434	0.196673	2.942058	-53170.9	-1.33	-94	-3797.92	0
36	3778.626	87.33327	49037.4	1987.184	660.1417	6526.274	672.3959	108.5126	133.2578	123.04	1.737599	8.347525	8.064959	20.94493	78.12856	8.8565	19.52479	52.22788	62.87611	96.7815	38.63132	4.105512	2.968917	2.540802	2.723419	4.029035	0.944225	-0.02596	0.927764	-0.07314	0.111177	0.16047	-0.04522	0.097873	2.813461	-255897	-1.64	-47	-2244.71	0
37	2653.984	14.72069	46202.25	7.095955	831.8707	17657.41	253.1614	41.18287	325.614	21.2267	3.56292	1.035746	5.014526	20.93675	78.70059	7.755615	19.85846	41.1023	67.49765	87.01124	9.700385	3.897917	3.066008	4.080146	4.933159	6.096778	0.874009	-0.09515	0.943901	-0.02283	0.095306	0.123629	-0.19299	0.185272	2.964268	-57260	-1.26	-94	-7157.5	0
38	212.0931	115.8598	44402.58	1515.731	633.5448	1334.149	262.7562	39.84287	635.3385	223.3579	4.033806	1.146827	9.977321	31.12582	89.62681	5.401451	12.59851	58.31952	65.10063	91.29174	7.441381	1.742499	3.079885	2.99156	4.794303	11.77225	0.865242	-0.16392	0.895679	-0.13592	0.141988	0.010406	-0.16479	0.068628	1.903535	-517446	-1.57	-94	-826.592	0
39	3955.561	137.5202	72282.54	1381.942	326.765	16641.33	65.50713	805.053	1116.195	118.5732	7.71694	4.020672	9.535722	24.96859	73.63751	5.639839	11.05527	41.32649	66.5784	98.54349	22.05829	3.717012	1.594498	4.860538	2.787943	5.357976	0.896076	-0.12951	0.997977	-0.01122	0.032575	0.05471	-0.11116	0.176412	1.233861	-6445.81	-2.66	-94	-6445.81	0
40	2859.684	25.29828	7571.042	439.8489	318.2346	1835.768	81.05306	997.7523	664.2644	106.5494	7.388722	5.662896	9.282844	35.15782	92.70692	8.043434	14.60621	57.93778	72.4253	99.81381	32.07095	1.036599	4.042025	1.069435	3.568564	6.380147	0.829626	-0.16355	0.850257	-0.06575	0.012769	0.113349	-0.19002	0.04949	2.386981	-910718	-1.19	-81	-2221.26	0
41	227.4916	156.5778	46501.11	1003.055	247.5393	1433.721	877.1741	213.2456	679.0451	227.3778	6.217549	1.146827	2.278558	44.09327	94.22263	6.526416	12.59851	51.02151	64.8904	91.52503	14.17118	1.69055	3.287151	4.772746	1.6788	11.60424	0.865242	-0.15648	0.893245	-0.13504	0.141988	0.014338	-0.05986	0.068628	1.894919	-703392	-1.42	-94	-926.735	0
42	443.5903	9.987807	71058.2	1840.317	268.1471	12726.92	559.4596	297.0395	946.8901	39.14747	7.849895	4.732427	6.721182	25.00363	81.25473	7.426688	26.88904	37.43678	74.75545	93.86413	4.061963	3.701607	2.697836	1.983312	2.745044	6.788245	0.864708	-0.10411	0.927815	-0.08118	0.080017	0.040274	-0.16407	0.023163	2.027282	-816120	-1.02	-94	-1961.83	0
43	3548.713	81.57444	24665.64	80.67773	109.3444	4111.553	897.6288	920.6461	601.9123	103.8851	9.463045	7.637372	7.327068	47.43473	86.93404	8.686092	12.12118	51.01363	68.44032	85.41322	20.19385	1.645043	3.988587	3.678514	2.722765	8.416535	0.986826	-0.1176	0.968922	-0.1981	0.148997	0.059765	-0.01482	0.180774	2.590606	-894268	-0.89	-94	-2378.37	0
44	2668.099	102.5478	46202.25	785.4904	776.9695	17976.56	408.8149	947.4759	268.9977	36.14241	3.56292	1.00554	5.014526	20.93675	80.17249	7.755615	19.85846	41.1023	72.24701	87.01124	9.700385	3.897917	3.17256	4.511355	4.933159	8.712872	0.882507	-0.09575	0.943901	-0.02283	0.095306	0.115977	-0.19397	0.035237	2.98558	-44705.4	-1.92	-94	-11176.3	0
45	1885.246	114.2869	47454.35	1773.674	509.3964	15482.7	983.0607	999.7669	972.2456	221.8573	4.351248	6.228239	2.006331	27.07525	73.21218	6.519776	11.43013	55.13604	78.23766	87.93192	13.61321	4.766555	2.672399	2.440069	2.494031	8.262777	0.81104	-0.04681	0.81348	-0.14218	0.103664	0.137912	-0.09867	0.172611	2.063899	-1207767	-0.92	-93	-3203.63	0
46	3818.925	106.3874	13057.02	814.3244	793.061	2565.066	882.6793	526.2759	272.0966	148.5817	7.936959	6.760296	9.656808	34.25904	60.82226	3.145358	13.36479	49.19933	67.67731	83.71384	11.26729	1.356065	4.102252	3.205442	1.587523	8.642076	0.801309	-0.10222	0.900406	-0.1077	0.11012	0.087477	-0.07931	0.106548	2.920382	-4910366	-0.77	-94	-7147.55	0
47	2357.339	145.9292	74420.32	1400.12	326.765	16645.59	313.3066	805.053	321.9629	221.7874	7.457616	3.917918	9.703346	44.08577	81.792	5.725002	25.49439	51.31277	67.86807	91.52028	15.67772	1.860401	1.594498	4.744034	1.740667	5.357976	0.967843	-0.02357	0.998356	-0.07439	0.056188	0.058821	-0.05307	0.050065	2.881619	-229389	-1.75	-94	-1674.37	0
48	1017.282	66.52784	49904.17	1535.485	40.14295	15463.25	768.3321	1048.355	841.1483	118.5157	4.365395	7.974286	9.968591	24.15249	82.26788	6.22751	20.77415	55.47726	67.30982	80.12451	27.60488	4.434676	2.536616	2.913044	2.115366	7.420004	0.851012	-0.04748	0.80492	-0.19065	0.139747	0.05732	-0.08636	0.071004	1.831265	-451360	-0.99	-94	-1872.86	0
49	977.8456	155.4394	54743.47	1083.92	460.5799	7641.597	823.9594	879.0883	174.3133	9.73905	7.506835	1.184566	9.588507	49.11151	57.86472	5.636573	26.94875	36.04617	63.79574	89.87094	5.268887	1.650591	1.069409	3.069217	3.537977	8.641437	0.885269	-0.13935	0.962276	-0.01228	0.120644	0.15874	-0.08609	0.116999	1.625214	-26019.2	-1.45	-94	-4336.54	0
50	257.9084	82.86486	72677.03	1988.961	196.2136	6598.759	101.7875	377.6704	754.9971	130.8637	1.140366	6.613251	5.727891	44.36726	75.34656	4.056631	26.26805	30.99138	77.94655	84.03523	5.010174	4.219697	2.160986	4.055664	2.87279	4.463102	0.871775	-0.02962	0.856399	-0.02738	0.093792	0.022174	-0.04777	0.155356	2.733343	-261264	-1.39	-91	-1016.59	0
51	3818.925	106.3874	20168.68	1594.009	793.061	669.5692	717.77	666.397	272.0966	147.8195	7.417606	6.760296	9.116176	34.25904	70.2154	3.145358	15.3662	49.34207	71.72734	88.6366	11.26729	1.356065	4.039993	3.082121	1.714325	6.362623	0.800763	-0.10222	0.900756	-0.15913	0.187728	0.087477	-0.0388	0.151275	2.936747	-592968	-0.75	-92	-1478.72	0
52	3484.02	159.2541	51081.51	1056.89	450.3322	15591.67	709.8492	715.549	157.6456	17.34821	7.438833	6.975926	2.495881	34.24354	76.90783	3.557275	25.63984	36.04617	63.33013	80.07345	5.604755	1.612433	1.657266	3.161826	2.088825	7.601554	0.806171	-0.16786	0.899047	-0.11789	0.125975	0.080202	-0.08609	0.116887	2.954241	-715581	-0.28	-94	-3726.99	0
53	1028.617	66.52784	49904.17	1593.466	991.5451	3874.692	768.3288	1002.344	854.4219	122.5496	8.489215	7.974286	9.901071	22.63646	82.41058	5.816124	20.77435	57.89802	67.30982	88.53322	14.03237	4.434676	2.498544	2.913044	2.115366	7.741784	0.813301	-0.04748	0.80492	-0.16177	0.107744	0.04533	-0.07629	0.064184	1.91689	-668774	-0.72	-94	-2685.84	0
54	1016.934	106.3874	20168.68	1513.19	0.431923	11338.38	765.4576	1097.876	130.8819	167.2353	4.467089	5.890095	9.656808	36.20809	80.67094	3.490032	20.98677	49.38839	67.99284	80.71645	38.85398	1.454099	2.157841	3.10541	1.756012	6.04093	0.822547	-0.10222	0.900756	-0.1998	0.190051	0.039845	-0.08861	0.153109	1.995201	-400015	-1.42	-94	-728.624	0
55	893.9919	181.5535	40864.18	1837.731	918.6841	8703.804	651.1371	1191.501	705.9739	91.26051	1.733023	6.350081	3.264958	47.56174	96.70397	9.621306	16.8604	42.90599	72.75573	95.00875	9.351469	1.755224	1.857347	2.662382	3.024162	10.01026	0.855851	-0.1006	0.906409	-0.01602	0.146076	0.185137	-0.17486	0.198588	1.302445	-54270.4	-1.1	-94	-2359.58	0
56	464.4197	143.6559	58752.22	1083.92	460.5799	15275.58	823.9594	877.2325	157.6456	3.804949	7.521033	7.961247	2.495881	48.19175	76.90783	3.944533	25.63984	36.04617	62.18555	82.64667	2.437069	1.644048	1.069409	4.353334	2.107389	7.083252	0.841325	-0.16444	0.962276	-0.11885	0.125975	0.142529	-0.08609	0.116999	1.5397	-114702	-1.54	-94	-1113.61	0
57	3818.925	106.3874	20168.68	1594.009	793.061	669.5692	717.77	666.397	272.0966	147.8195	8.407549	6.760296	9.656808	34.25904	59.53921	3.145358	15.3662	49.34207	67.83305	88.6366	11.26729	1.356065	4.039993	3.082121	1.714325	6.362623	0.800763	-0.10222	0.900756	-0.15913	0.192595	0.087477	-0.0388	0.151275	2.936747	-1251053	-0.35	-94	-2415.16	0
58	1044.478	86.95037	24713.96	1876.993	109.3444	3242.042	744.9001	323.0552	607.4948	78.93014	7.146251	8.357373	8.859216	23.71881	99.96221	7.667927	17.1162	50.62047	62.33869	98.27191	23.82969	1.645043	3.115138	2.818348	4.794176	10.13806	0.80265	-0.03242	0.931512	-0.19193	0.111453	0.060091	-0.05239	0.179725	2.732296	-831276	-0.62	-94	-3338.46	0
59	2859.684	25.29828	16640.71	439.8489	318.2346	1835.768	81.05306	997.7523	664.2644	207.1494	7.388722	5.662896	9.282844	35.15782	92.70692	8.043434	14.60621	57.93778	72.4253	99.81381	32.07095	1.036599	4.046433	1.069435	3.568564	6.380147	0.829626	-0.16355	0.850257	-0.06575	0.012769	0.085542	-0.19002	0.047463	2.386981	-1105643	-0.86	-90	-2484.59	0
60	3013.235	153.9145	42698.09	1964.31	312.5208	11050.83	718.5713	675.9349	704.7513	104.4245	6.220704	9.888197	3.34304	25.6791	52.53824	9.017014	11.39876	33.88617	77.528	99.57248	2.77593	2.888656	4.354758	4.146533	3.998091	4.091477	0.984704	-0.10497	0.81584	-0.16226	0.044914	0.048555	-0.02231	0.08318	2.470483	-427363	-0.76	-94	-2374.24	0
61	117.6267	7.448917	38097.98	519.7123	777.7647	11619.41	141.5263	927.6605	408.3622	119.6793	8.876761	8.157838	4.971717	47.536	74.14062	2.301829	25.97787	48.6498	66.43854	96.14959	25.99743	1.154918	4.264424	3.67572	1.346461	8.452089	0.801328	-0.03282	0.904707	-0.13389	0.144423	0.163339	-0.15713	0.16389	2.666624	-498254	-0.96	-94	-1798.75	0
62	947.8275	182.6966	12750.8	1915.014	923.3605	15343.82	671.6974	567.6117	878.3842	86.19088	2.142686	1.085401	3.409004	40.91943	85.0636	9.861331	20.49793	49.42201	73.56275	85.28602	12.91196	3.968317	2.59315	2.570902	2.521973	8.759155	0.876421	-0.18585	0.944584	-0.02138	0.024586	0.140698	-0.19468	0.156775	1.431156	-68000	-0.85	-94	-6800	0
63	1629.307	1.109168	39386.88	1559.359	345.0612	15725.83	159.1288	79.04372	823.9734	2.086337	1.246921	3.380563	9.205887	22.68229	90.68988	8.967738	24.34897	31.91425	76.86734	86.6972	22.86469	3.973863	1.1221	4.05278	1.559619	5.795006	0.970933	-0.05961	0.876975	-0.10323	0.13573	0.165976	-0.18944	0.140291	2.267756	-342923	-0.87	-94	-2700.18	0
64	2210.53	160.7202	32529.33	1461.701	425.3646	2907.985	10.79362	770.1665	751.199	66.70382	5.311058	7.134329	7.509188	27.32435	83.49295	7.627033	26.78223	35.07665	79.43141	86.08465	28.87095	1.890292	4.987977	3.223555	1.909661	7.95297	0.957733	-0.11791	0.800965	-0.01691	0.112845	0.174362	-0.04928	0.096805	1.725948	-40305.7	-1	-94	-283.843	0
65	200.6743	77.2734	44371.76	1131.742	633.5448	340.0537	270.6216	39.84287	802.9205	222.4241	9.985221	1.146827	3.91359	31.12582	89.62681	5.401451	21.03589	58.31952	67.65503	91.63423	16.39148	1.866663	3.079885	3.98295	4.471606	11.86219	0.864759	-0.16392	0.896695	-0.0411	0.148889	0.010808	-0.16612	0.068628	1.89391	-202790	-0.68	-94	-321.379	0
66	1116.316	182.6699	18194.3	30.48446	539.9713	5630.899	215.1279	941.7383	815.6707	51.25282	4.707105	1.640246	5.398812	47.14139	59.96623	5.163945	13.54548	46.83841	71.17252	91.73728	37.6638	1.566047	2.450198	2.887984	3.601172	8.717394	0.884481	-0.15934	0.977036	-0.01661	0.124729	0.062381	-0.1993	0.079878	2.413089	-84416.4	-0.86	-94	-511.615	0
67	2939.752	93.66816	22521.91	1156.57	315.0607	2664.467	270.6746	577.2666	751.199	99.38004	6.352801	6.46343	9.951286	35.52266	50.85143	8.023344	24.01355	34.20274	70.98267	86.08465	42.66108	1.224326	4.99486	3.153087	2.483266	6.088613	0.984906	-0.1163	0.809642	-0.03125	0.112845	0.175095	-0.04928	0.096805	1.725948	-89638.1	-0.75	-94	-207.496	0
68	3348.083	118.5943	1997.084	896.2985	246.5829	13629.67	272.3231	254.7841	74.22392	126.1307	6.603338	9.559784	8.012419	35.61557	57.85638	7.980374	28.01796	56.35243	66.68091	81.2429	35.32338	4.925748	3.017794	1.721259	3.375519	4.611724	0.874575	-0.11508	0.853169	-0.01234	0.026029	0.025918	-0.07358	0.084043	2.489178	-34281	-1.15	-73	-300.711	0
69	3548.032	92.03763	49037.4	859.7043	644.7782	6526.274	860.8709	108.5126	146.0419	127.133	1.737599	8.58722	7.745017	20.94493	76.52531	5.835251	26.38329	52.25149	66.51733	93.86273	40.50245	4.105512	2.968917	2.612691	2.699555	4.420548	0.953414	-0.01752	0.927253	-0.07314	0.065843	0.177611	-0.08406	0.108986	2.813461	-16295.1	-0.32	-94	-4073.78	0
70	101.8593	19.15446	43024.09	1483.465	460.6351	12603.53	146.297	242.4121	1117.02	104.1606	1.073974	3.136199	6.629404	38.08221	54.29554	6.931697	25.95331	49.24994	64.54847	92.54805	26.64733	2.769504	1.97263	3.287291	2.855044	11.78177	0.986489	-0.16649	0.847117	-0.12307	0.198551	0.07018	-0.02256	0.197422	2.432643	-506789	-0.61	-94	-2184.43	0
71	1980.429	114.2869	47454.35	504.3707	630.2667	14220.26	947.8774	999.7669	972.2456	221.8573	6.178319	6.228239	2.006331	28.36177	73.21218	6.519776	11.43013	51.68497	77.67146	87.14313	13.61321	1.692351	2.67733	2.440069	2.494031	8.308005	0.81104	-0.11745	0.81348	-0.0681	0.103664	0.137912	-0.10053	0.172611	2.063899	-721661	-0.72	-89	-1694.04	0
72	117.6267	7.448917	38097.98	169.3028	899.2626	11619.41	141.5263	927.6605	408.3622	119.6793	8.876761	8.433377	4.971717	47.536	74.14062	2.301829	26.55809	48.6498	66.43854	93.59553	20.50592	1.154918	4.264424	3.67572	1.346461	8.452089	0.801328	-0.03282	0.904707	-0.13389	0.144423	0.163339	-0.15713	0.16389	2.666624	-361466	-0.65	-94	-1304.93	0
73	486.721	12.92082	82007.83	460.7653	676.7918	8895.746	627.5123	107.5045	310.6478	226.6969	4.318724	3.353186	2.429782	42.54249	60.75261	6.6572	12.77627	37.66824	74.93406	92.98877	4.311109	3.765339	4.9707	2.096315	3.663108	6.779028	0.864939	-0.17491	0.953751	-0.08039	0.074782	0.036249	-0.16362	0.025137	2.325705	-647727	-0.42	-92	-1709.04	0
74	949.8409	172.7115	3945.679	1837.731	918.6841	6344.761	638.6158	1191.501	662.9062	129.7266	1.878881	6.350081	3.264958	47.56174	94.74363	9.863731	15.94283	42.90599	72.75573	95.00875	11.70819	1.755224	1.857347	2.662382	3.352996	9.700997	0.877283	-0.11718	0.906409	-0.01709	0.146076	0.014794	-0.19605	0.198588	2.431762	-156548	-0.7	-94	-2371.94	0
75	1017.282	66.52784	49904.17	1543.447	40.14295	15436.28	695.8771	1153.198	841.1483	118.5157	4.365395	7.974286	9.901071	24.15249	82.26788	6.22751	20.77415	49.41496	67.30982	80.12451	36.29485	4.434676	2.536616	2.913044	2.115366	7.420004	0.818337	-0.04748	0.80492	-0.19065	0.121302	0.063719	-0.08778	0.070315	1.831265	-299103	-0.57	-94	-1278.22	0
76	1237.074	77.80713	27955.04	548.4449	451.4873	808.601	288.8751	437.6095	872.0107	22.47845	3.629744	5.200599	3.815192	37.11406	65.32019	7.61885	24.12085	30.42988	68.60877	95.86811	38.68098	3.633862	2.907545	4.73705	2.39257	7.526419	0.974299	-0.04336	0.924084	-0.19869	0.118175	0.028321	-0.18348	0.101395	1.357989	-39610.4	-0.72	-94	-129.446	0
77	1764.245	173.0519	57931.32	749.3678	467.4649	2715.898	24.54012	778.9936	355.1501	83.24718	5.133285	8.248943	6.438948	24.57124	57.76551	6.537638	26.055	41.53184	79.35823	95.49905	27.11829	4.283283	4.96767	4.267912	2.778	6.558415	0.834039	-0.13363	0.811457	-0.03243	0.087975	0.175206	-0.07053	0.081601	1.825462	-137173	-0.47	-94	-1428.89	0
78	838.1469	27.32987	28366.18	754.5115	20.79689	5317.183	873.1757	103.4454	1019.728	200.8193	6.716531	1.599766	9.923791	31.12004	81.88421	5.301751	17.7451	34.81091	75.62726	82.80259	48.15828	4.63542	1.323689	3.252889	1.911591	10.03011	0.833089	-0.17832	0.80581	-0.13865	0.166454	0.097188	-0.07674	0.183663	2.845893	-381291	-0.15	-43	-1406.98	0
79	3548.032	92.03763	35617.73	1223.421	644.7782	6236.069	860.8709	108.5126	138.3566	127.133	1.737599	8.58722	7.745017	20.94493	76.52531	5.835251	26.38329	52.25149	66.51733	95.59408	40.50245	3.32392	2.968917	2.612691	2.699555	4.420548	0.953414	-0.0131	0.927253	-0.07314	0.065843	0.177611	-0.08406	0.108986	2.721275	-14823.7	-0.31	-94	-2964.74	0
80	1683.748	87.66769	27442.2	1350.242	425.3646	898.4466	270.6746	770.1665	751.199	85.87995	5.311058	7.106013	3.79917	39.79053	55.99845	7.627033	24.01355	34.20274	79.43141	86.08465	40.61657	1.224326	4.987977	3.153087	2.435437	7.95297	0.984906	-0.1163	0.800965	-0.03125	0.112845	0.175095	-0.04928	0.096805	1.725948	-101633	-0.65	-94	-377.817	0
81	3685.186	88.45808	98886.88	475.217	99.63019	5235.373	107.1285	67.63628	581.3697	193.0898	8.599589	2.289788	3.856011	49.22266	75.46677	5.945809	25.13928	38.91359	62.02217	87.54152	33.41615	1.86237	2.523353	4.962017	1.423659	9.141977	0.889524	-0.13134	0.970359	-0.18626	0.039162	0.119441	-0.18961	0.180105	1.809872	-319737	-0.47	-94	-630.646	0
82	22.92808	124.8942	46501.11	445.0457	633.5448	1433.721	870.311	6.911922	635.3385	227.3778	6.217549	1.146827	2.127714	31.01651	94.22263	6.335486	12.59851	58.8542	65.10063	91.29174	13.40279	1.69055	3.079885	2.99156	4.794303	11.60424	0.865242	-0.16392	0.895235	-0.13592	0.141988	0.010406	-0.03507	0.068628	1.894919	-110331	-0.62	-94	-184.5	0
83	3908.273	54.14313	45924.52	1594.009	793.061	2420.388	717.77	746.5455	272.0966	147.8195	6.276494	6.335782	9.674478	30.98047	59.53921	6.430636	13.34837	58.17913	65.44486	86.47445	11.26729	1.637735	3.038723	3.092544	1.587523	11.41588	0.866939	-0.16268	0.895753	-0.16573	0.192595	0.014547	-0.0507	0.130054	1.789526	-487583	-0.23	-94	-843.569	0
84	1968.732	112.5859	28294.09	504.3707	630.2667	14220.26	947.8774	999.7669	1003.866	221.8573	5.308002	5.401426	2.372198	29.57444	55.44335	7.550541	13.37977	51.4367	77.67146	87.41612	37.31847	1.571312	2.214237	2.440069	2.494031	8.308005	0.826063	-0.11625	0.800959	-0.0681	0.103664	0.137912	-0.10053	0.172611	1.667605	-264966	-0.36	-94	-1292.52	0
85	1116.316	182.6699	18194.3	31.707	539.9713	5753.632	277.7677	157.9247	815.6707	44.12433	4.707105	1.201463	5.590037	37.43459	59.96623	5.163945	13.36456	46.83841	67.02632	90.07372	37.6638	4.41146	2.586342	2.627074	3.601172	8.757554	0.884481	-0.15934	0.977036	-0.01661	0.124729	0.062381	-0.1872	0.079626	2.413089	-127435	-0.5	-94	-2275.63	0
86	502.5382	9.987807	71263.88	1869.803	255.7278	16998.39	544.6085	316.9451	946.8901	141.0027	3.875281	4.521008	6.721182	25.00363	80.78626	7.780686	20.8535	38.30153	74.75545	94.29836	4.061963	3.743122	3.261237	3.953749	2.745044	11.35258	0.864708	-0.10411	0.834387	-0.04028	0.080017	0.098292	-0.16407	0.023163	2.019791	-173729	-0.34	-91	-1165.97	0
87	3955.561	165.2742	69699.27	1381.942	326.765	16641.33	402.6087	819.9106	1093.284	86.59239	7.71694	3.927248	9.50387	40.85902	73.63016	5.65982	23.76188	43.39801	67.86807	98.54349	22.05829	3.717012	1.594498	4.744789	2.689629	5.555103	0.967843	-0.173	0.998356	-0.01122	0.070927	0.05471	-0.10109	0.176412	2.91713	-1881.69	-0.23	-94	-1881.69	0
88	1629.307	1.504408	36195.32	1559.359	345.0612	15905.9	541.2843	79.04372	823.9734	54.70876	1.357285	3.810989	9.636725	22.68229	64.86006	7.563282	24.4136	34.50946	71.99263	97.45774	22.86469	3.826771	1.240622	1.900957	1.559619	5.795006	0.845935	-0.0556	0.87963	-0.10323	0.13573	0.165976	-0.08885	0.020922	2.132716	-114649	-0.39	-94	-703.368	0
89	3955.561	137.5202	69699.27	1381.942	326.765	16641.33	402.6087	805.053	1093.284	86.59239	7.71694	3.927248	9.50387	40.85902	73.63751	5.65982	23.76188	41.32649	67.86807	98.54349	22.05829	3.717012	1.594498	4.744034	2.689629	5.357976	0.967843	-0.173	0.998356	-0.01122	0.056188	0.05471	-0.10109	0.176412	2.91713	-1881.69	-0.23	-94	-1881.69	0
90	3877.693	92.03763	49037.4	859.7043	660.1417	6526.274	639.6083	108.5126	146.0419	127.133	1.737599	8.58722	7.745017	20.94493	76.52531	5.835251	26.38329	52.25149	66.51733	93.86273	40.50245	4.105512	2.968917	2.540802	2.699555	4.420548	0.943729	-0.01752	0.927253	-0.07314	0.065843	0.177611	-0.08406	0.097873	2.813461	-7876.67	-0.16	-94	-1312.78	0
91	540.2545	93.01583	97379.92	1427.669	270.3981	12199.97	80.58999	910.3708	261.3762	102.7001	3.717919	5.741359	7.840019	38.55095	50.4002	2.885156	16.402	34.03499	69.16944	85.53773	11.00289	1.139434	2.932417	1.667638	2.777897	4.357	0.920622	-0.03858	0.968479	-0.16984	0.195989	0.169471	-0.15057	0.19561	2.949649	-175364	-0.3	-94	-1169.1	0
92	1017.282	66.52784	49904.17	1535.485	40.14295	11193.06	768.4772	1153.198	841.1483	118.5157	4.365395	7.974286	9.901071	24.15249	82.26788	6.22751	20.77415	55.47726	67.30982	80.12451	27.60488	4.434676	2.536616	2.913044	2.115366	7.420004	0.818337	-0.04748	0.80492	-0.19065	0.139747	0.05732	-0.08636	0.071004	1.831265	-216166	-0.31	-94	-896.954	0
93	632.8516	108.4906	39721.37	854.4452	531.7925	252.1185	405.7083	960.3747	544.6484	206.6566	7.766588	3.838991	4.952365	37.48327	96.373	2.660757	23.70676	41.58542	72.71933	88.86741	5.01704	4.687802	3.598408	1.676813	2.526848	8.606635	0.868867	-0.03574	0.806294	-0.10472	0.109653	0.02708	-0.09369	0.105536	2.645657	-252348	-0.09	-45	-489.996	0
94	227.4916	156.5778	46501.11	979.5782	246.9242	1433.721	877.1741	213.2456	1196.15	89.96891	6.217549	1.937014	2.278558	43.10897	94.22263	6.461234	12.59851	42.7698	63.96422	91.43954	14.17118	2.97739	3.511699	4.99778	4.125895	11.60424	0.865242	-0.15648	0.900995	-0.13692	0.141988	0.014338	-0.11586	0.165387	1.894919	-232666	-0.11	-94	-538.578	0
95	3592.301	106.3155	97433.88	1449.28	656.2215	2599.892	258.7307	737.2536	261.5858	63.27457	9.245244	4.602626	6.486002	24.62251	62.82657	9.421883	19.011	57.33968	79.56358	94.80902	21.80198	1.542112	4.280239	2.045219	3.283938	5.200391	0.888311	-0.09588	0.935765	-0.09313	0.072067	0.089954	-0.16005	0.196259	1.776416	-99516.9	-0.52	-94	-398.067	0
96	3626.567	88.45808	98886.88	452.6777	97.38495	5147.788	953.0507	0.31497	576.4584	193.0898	8.575141	2.135324	3.940601	44.52025	75.46677	6.270414	25.50774	38.9742	62.02217	87.38926	31.54148	1.895156	2.597088	4.777821	1.482189	9.141977	0.889524	-0.10485	0.942321	-0.18626	0.038838	0.119441	-0.18961	0.180105	1.809872	-205665	-0.45	-94	-667.743	0
97	3626.567	88.45808	98886.88	452.6777	97.38495	5147.788	953.0507	0.31497	576.4584	193.0898	8.818969	2.268513	3.940601	44.52025	75.46677	6.270414	26.873	38.9742	62.02217	87.38926	38.10975	1.895156	3.290342	4.777821	1.482189	9.141977	0.889524	-0.15581	0.942321	-0.18626	0.038838	0.137921	-0.18961	0.180105	1.809872	-200724	-0.42	-94	-662.457	0
98	1017.282	66.52784	49904.17	1543.447	40.14295	15436.28	768.4772	1153.198	841.1483	118.5157	4.365395	7.974286	9.901071	24.15249	82.26788	6.22751	20.77415	55.47726	67.30982	80.12451	36.29485	4.434676	2.536616	2.913044	2.115366	7.420004	0.818337	-0.04748	0.80492	-0.19065	0.107744	0.05732	-0.07629	0.073616	1.831265	-168048	-0.24	-94	-694.415	0
99	464.4197	56.60662	58752.22	1083.92	460.5799	14395.12	178.8141	877.3407	157.6456	62.68086	7.694194	7.961247	2.495881	44.41693	75.24301	5.715061	24.76958	36.04617	62.79113	82.54688	5.268887	1.569382	1.331408	4.392733	2.116606	7.083252	0.817137	-0.16444	0.915105	-0.1149	0.125975	0.166857	-0.08609	0.116348	1.625214	-68212.8	-0.4	-94	-631.6	0
100	1044.478	81.57444	23032.31	1876.993	109.3444	4111.553	919.9329	323.0552	607.4948	78.93014	9.008881	8.357373	8.859216	44.58144	95.78441	8.467634	17.41236	50.97675	62.33869	85.43618	23.30706	1.645043	2.050379	4.054657	4.794176	6.789762	0.938002	-0.10703	0.931512	-0.19997	0.113747	0.060091	-0.04917	0.179725	2.732296	-200225	-0.13	-94	-650.083	0

Like Table 1. Shows, after adjustments, we got the 100 sets of top performance parameters. By observation, we found that some parameters regressed around certain enumerated values, so we directly take their plural for these parameters as our final parameters. For those parameters with more dispersed enumeration values, we took the average value of them as our final choice. Finally, as Graph 2 demonstrated, we achieved a robust profit in the test set (last 500 days part 2 data) which had a relatively low drawdown and relatively stable profit over time.

Graph 2. Performance of Combined Strategy with optimized parameter set on Part 2 data



2.Justification of submitted strategy

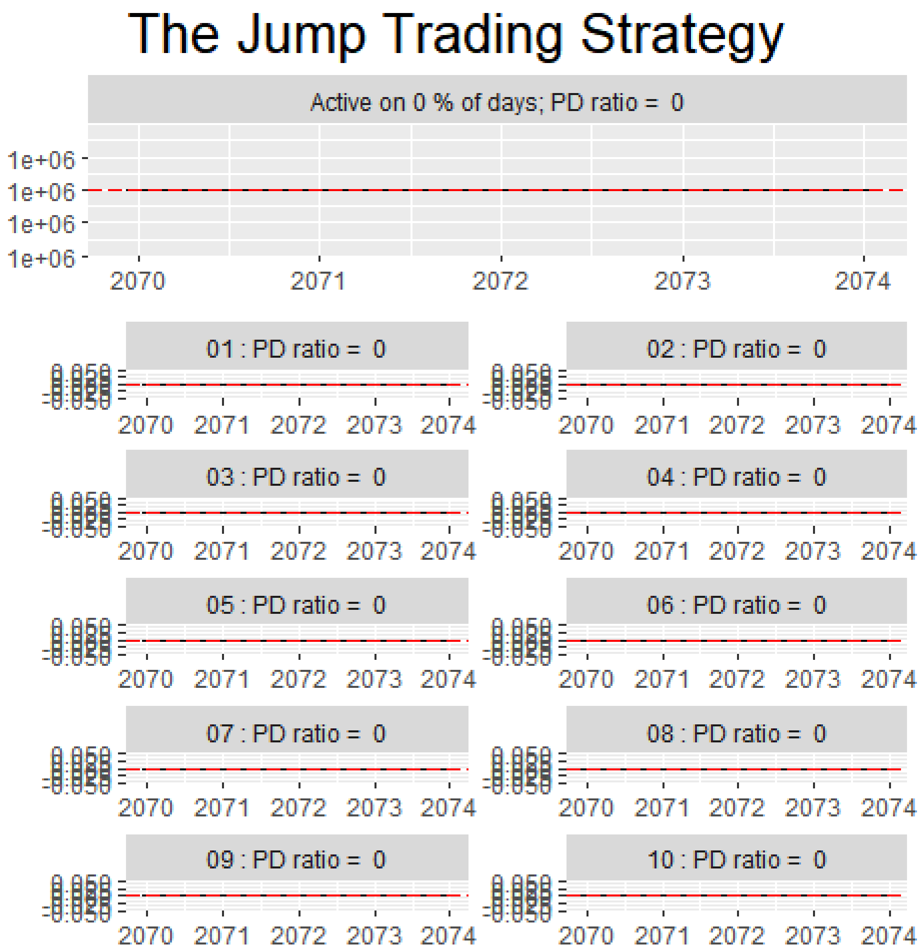
2.1 The process led to this particular strategy

In part 2 of the project, from data analysis, 6 strategies are developed that might be suitable for trading 10 series. However, as can be seen from Graph 3, in the test set (1000 days part1 data + 500 days part2 data), we found that The Jump Trading Strategy did not trigger at all no matter what parameters we chose, so we just discarded it.

```r
# 4. The Jump Trading Strategy ------------------------------------------
# (abandoned)
#
# The Jump Trading System is a psychological trading system that
# measures sudden price changes due primarily to excessive emotional reactions.
# Its basic movement is in a downtrend where prices fluctuate
# around the lower range of the box range for 5 to 10 days,
# then open price sharply lower below the trend line with extreme selling sentiment,
# and if it then rebounds to yesterday's lows, it indicates a reversal of market energy
# and another bullish upward move is ready to take place. Systematic buying mechanical rules.
# 1. close price 4% below the five-day MA to ensure the signal occurs on a downtrend.
# 2. and open price is 1% below yesterday's low price.
# 3. Closing rally above yesterday's low price.

# init params
params.jt <- list(lookback=5, series=1:10, posSizes=rep(1,10), holdPeriod=6) # tbd

# main strategy
getOrders.jt<- function(store, newRowList, currentPos, info, params){
  # used for initializing vectors
  allzero  <- rep(0,length(newRowList))

  # init store
  if (is.null(store)) {
    store <- initStore(newRowList,params$series)}else{
      store <- updateStore(store, newRowList, params$series)
    }

  # init today's enter positions on each series
  pos <- allzero

  # default exit yesterday's overall positions
  marketOrders <- -currentPos

  # we need to wait until day 101 so that we have ema100 signal
  if(store$iter > params$lookback){
```

```r
    for(i in params$series){
      # everyday re-calculate today's ema5
      ema5 <- last(EMA(store$cl[1:store$iter,i],5))

      # long enter
      if(((store$cl[store$iter,i] - ema5) / ema5 <= -0.04) &
         ((store$op[store$iter,i] - store$lo[store$iter-1,i]) / store$lo[store$iter-1,i] <= -0.01) &
         (store$cl[store$iter,i] > store$lo[store$iter-1,i])){
        # pos[i] <- params$posSizes[i]*1000000 / store$cl[store$iter,i]
        pos[i] <- params$posSizes[i]

      # exit when direction turned again
      }else if(((currentPos[i] != 0) &
                (store$cl[store$iter,i] < store$cl[store$iter-1,i]))){
        pos[i] <- 0

      # hold when still in the trend
      }else if(((currentPos[i] != 0) &
                (store$cl[store$iter,i] >= store$cl[store$iter-1,i]))){
        pos[i] <- currentPos[i]
      }
      #
      # # check if we have been in trade too long
      #
      # # we maintain that pos[i] is an integer
      # # if pos[i] == 0 we were flat last period
      # # if pos[i] >  0 we have been long  for store$count[i] periods
      # # if pos[i] <  0 we have been short for store$count[i] periods
      #
      # if (pos[params$series[i]] > 0) {# long signal today
      #   if (store$count[i] < 0) # last time we were short
      #     store$count[i] == 1
      #   else if (store$count[i] == params$holdPeriod) { # reached holding period
      #     pos[params$series[i]] <- 0 # don't stay long
      #     store$count[i] <- 0 # reset count to 0
      #   }
      #   else # 0 <= store$count[i] != (should be <) params$holdPeriod
      #     store$count[i] <- store$count[i] + 1
      # }
      #
      # else if (pos[params$series[i]] < 0) {# short signal today
      #   if (store$count[i] > 0) # last time we were long
      #     store$count[i] == -1
      #   else if (store$count[i] == -params$holdPeriod) { # reached holding period
      #     pos[params$series[i]] <- 0 # don't stay short
      #     store$count[i] <- 0 # reset count to 0
      #   }
      #   else # 0 >= store$count[i] != (should be >) -params$holdPeriod
      #     store$count[i] <- store$count[i] - 1
      # }
      # else{
      #   store$count[i] <- 0 # reset count to 0
      # }
    }
  }
  marketOrders <- marketOrders + pos
  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=allzero,limitPrices1=allzero,
              limitOrders2=allzero,limitPrices2=allzero))
}

test.jt<- backtest(dataList, getOrders.jt, params.jt, sMult=0.2)
pfolioPnL.jt <- plotResults(dataList, test.jt, plotType='ggplot2',
                            titleString='The Jump Trading Strategy')
```

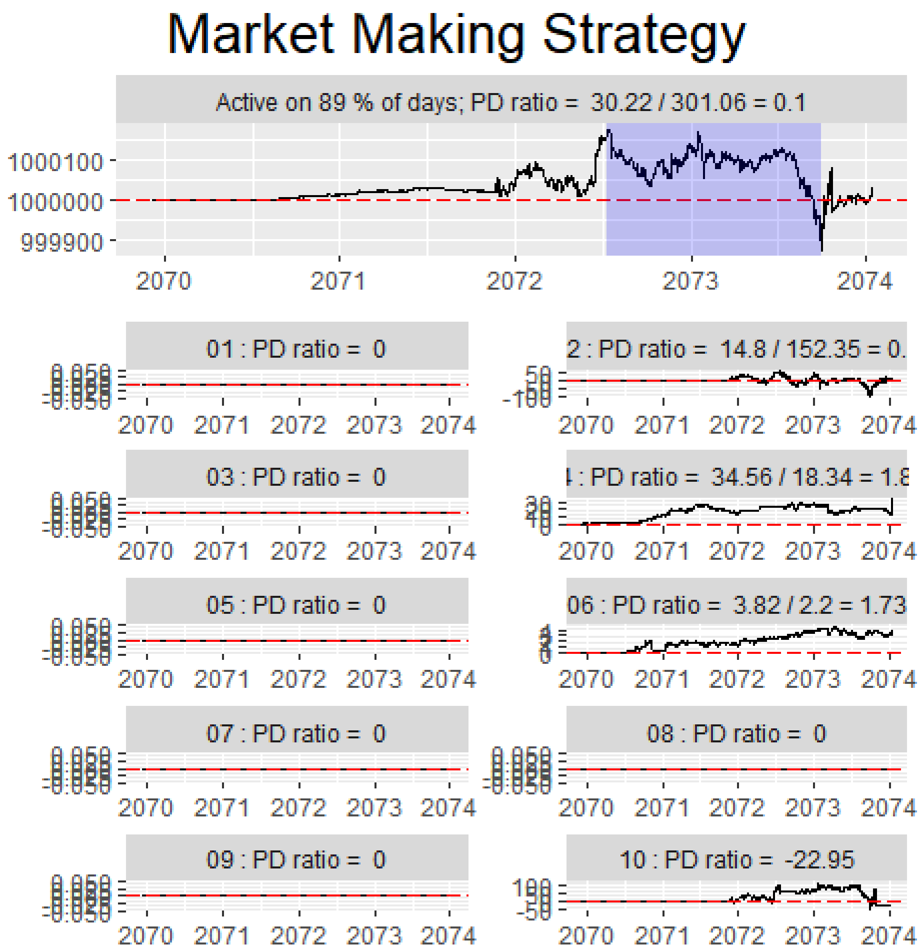
Graph 3. Performance of The Jump Trading Strategy with default parameters on training set (1000 days part 1 + 500 days part 2)



The Market Making Strategy performed well in both part 1 and part 2 data like Graph 3 displayed, but due to the immaturity of our system design for trading signal logging mechanism before development, we mistakenly used the Market Orders information sent back to the Backtester framework for each day to record trade information. To make it worse, our initial code was developed with BBands Mean-reversion Market Orders, which led to a redesign of the entire signal system if we tried to use Limit Orders later on, so we finally had to abandon the Market Making Strategy and Limit Orders.

```r
# 3. Market Making Strategy ---------------------------------------------
# One of the most important types of high-frequency trading strategies is
# Market Making, i.e., making profits by earning bid-ask spreads
# But in our case, we want to exploit the market by setting a proper
# spread range so that we can betting on both sides by limit orders
# Buy low and sell high to make profit

# init params
params.mm <- list(lookback=0, series=1:10,
                  posSizes=rep(1,10), spread=0.08, holdPeriod=6) # tbd

# main strategy
getOrders.mm<- function(store, newRowList, currentPos, info, params){
  # used for initializing vectors
  allzero  <- rep(0,length(newRowList))

  # if our positions reach 2 times betting level then exit yesterday's overall positions
  marketOrders<- ifelse(abs(currentPos)>2*params$posSizes, -currentPos, 0)

  # init store
  if (is.null(store)) {
    store <- initStore(newRowList,params$series)}else{
      store <- updateStore(store, newRowList, params$series)
    }

  # init today's enter positions on each series
  pos <- allzero

  # init limitPrices1/limitPrices2/limitOrders1/limitOrders2
  limitPrices1<- allzero; limitOrders1<- allzero
  limitPrices2<- allzero; limitOrders2<- allzero

  if(store$iter > params$lookback){
    for(i in params$series){
      # market making - betting on both sides
      # limitOrders1[i]<- params$posSizes[i]*1000000/store$cl[store$iter,i]
      limitOrders1[i]<- params$posSizes[i]
      limitPrices1[i]<- (store$hi[store$iter,i]+store$lo[store$iter,i])/2-params$spread*store$cl[store$iter,i]
      limitOrders2[i]<- -params$posSizes[i]
      limitPrices2[i]<- (store$hi[store$iter,i]+store$lo[store$iter,i])/2+params$spread*store$cl[store$iter,i]
      #
      # # check if we have been in trade too long
      #
      # # we maintain that pos[i] is an integer
      # # if pos[i] == 0 we were flat last period
      # # if pos[i] >  0 we have been long  for store$count[i] periods
      # # if pos[i] <  0 we have been short for store$count[i] periods
      #
      # if (pos[params$series[i]] > 0) {# long signal today
      #   if (store$count[i] < 0) # last time we were short
      #     store$count[i] == 1
      #   else if (store$count[i] == params$holdPeriod) { # reached holding period
      #     pos[params$series[i]] <- -currentPos[i] # don't stay long
      #     store$count[i] <- 0 # reset count to 0
      #   }
      #   else # 0 <= store$count[i] != (should be <) params$holdPeriod
      #     store$count[i] <- store$count[i] + 1
      # }
      #
      # else if (pos[params$series[i]] < 0) {# short signal today
      #   if (store$count[i] > 0) # last time we were long
      #     store$count[i] == -1
      #   else if (store$count[i] == -params$holdPeriod) { # reached holding period
      #     pos[params$series[i]] <- -currentPos[i] # don't stay short
      #     store$count[i] <- 0 # reset count to 0
      #   }
      #   else # 0 >= store$count[i] != (should be >) -params$holdPeriod
      #     store$count[i] <- store$count[i] - 1
      # }
      # else{
      #   store$count[i] <- 0 # reset count to 0
      # }
    }
  }
  marketOrders <- marketOrders + pos
  return(list(store=store,marketOrders=marketOrders,
              limitOrders1=limitOrders1,limitPrices1=limitPrices1,
              limitOrders2=limitOrders2,limitPrices2=limitPrices2))
}

test.mm<- backtest(dataList, getOrders.mm, params.mm, sMult=0.2)
pfolioPnL.mm <- plotResults(dataList, test.mm, plotType='ggplot2',
                            titleString='Market Making Strategy')
```

Graph 4. Performance of Market Making Strategy with default parameters on training set (1000 days part 1 + 500 days part 2)



Regarding the selection of trading strategies and series, we found that with the initial parameter settings, the remaining 4 trading strategies were able to obtain different profit on different series. But each strategy itself triggered not much, only being active 10-20% of period. So, it was natural to come out with the idea to involve them in the decision making together. However, when we noticed a problem that if all strategies make decisions together, multiple strategies may be triggered on the same series at the same time, and the positions placed by different strategies may lead to additional positions or cancel each other out. Therefore, we decided to set priorities according to the performance of the four strategies in the test set, so that the strategy with the higher priority makes the first decision. If the strategy with the higher priority does not place any order, then the strategy with the lower priority continues to make judgments until all four strategies complete their decisions. Finally, we execute the highest priority non-empty trade order.

```r
# part of code from Combined Strategy code

    # market condition filter(tbc)
    # 1. if trigger anomoly(e.g. extreme low volume...) then we stop trading for at least 30 days(tbd)
    # 2. ...

    bbtf <- getOrders.bbtf(store, newRowList, currentPos, info, params)
    tma <- getOrders.tma(store, newRowList, currentPos, info, params)
    bbmr <- getOrders.bbmr(store, newRowList, currentPos, info, params)
    lmv <- getOrders.lmv(store, newRowList, currentPos, info, params)

    # order by Priority (derived from individual Strategy Performance)
    orders_4 <- list(bbtf, tma, bbmr, lmv)

    # decide which strategy should dominate

    for(i in params$series){  # looping series

      # main logic

      if(currentPos[i] == 0 & flagStore[store$iter,i] == 6){
        order_index <- 1
        # looping orders order by Priority
        for(s in orders_4){
          # if orders[s] decides to long or short any shares of stock using market/limit orders
          # then stop and output orders to combined strategy orders
          if(s$marketOrders[i] != 0){
            marketOrders[i] <- s$marketOrders[i]

            # store which strategy is dominated which series everyday, store in codes
            # 1:5 stands for (mm, bbtf, tma, bbmr, lmv) seperately
            SOCStore[store$iter+1,i] <<- order_index
            break
          }
          order_index <- order_index + 1
        }
      }

      else if(currentPos[i] != 0){  # if someone is on charge
        # continue using decisions made by the same strategy
        SOCStore[store$iter+1,i] <<- SOCStore[store$iter,i]
        marketOrders[i] <- orders_4[[SOCStore[store$iter+1,i]]]$marketOrders[i]
      }
    }
```

In relation to the selection of series, the Kelly Formula is utilized based on the performance of the 10 series on the test set to get the long-term optimal wager, which ensures optimal long-term profitability with minimal risk of bankruptcy.

2.2 Justifying the choice of position sizing, and other key elements of strategy.

As mentioned in the Part 1 report, there are three main components in our position sizing strategy: Wager strategy, Capital Utilization and Leverage.

According to the Wager Strategy, we tried the Kelly formula with mathematically strict long-term optimal betting ratios, the Martingale Strategy and the Reverse Martingale strategy respectively, using the BBands Mean-reversion Strategy with default parameter settings. For comparison, we used fixed wager strategy and performance-based fixed wager strategy as comparison groups as benchmark, and we also split the training and test sets to observe the performance of these 3 strategies over time to ensure our results are robust.

```r
# **********************************
# Martingale Wager strategy

Martingale <- function(params, store){
  for (i in params$series){
    # store money spend for establish position
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      Martingale_lastProfit[i] <<- -moneySpendStore[store$iter+1, i]
    }
    # when clear position, calculate this trade whether lose or win,
    # when losing, lastProfit = 1 , else 0
    else if(flagStore[store$iter,i] == 5){
      if (Martingale_lastProfit[i] - moneySpendStore[store$iter+1, i] < 0){
        Martingale_lastProfit[i] <<- 1
      }else{
        Martingale_lastProfit[i] <<- 0
      }
    }
    # when clear position and lose, double the wager for the next time
    if (flagStore[store$iter,i] == 5 & Martingale_lastProfit[i] == 1){
      Martingale_multiplier[i] <<- Martingale_multiplier[i] * params$Martingale
    }
    # if win, just reset the wager to the initial amount
    else if (flagStore[store$iter,i] == 5 & Martingale_lastProfit[i] == 0){
      Martingale_multiplier[i] <<- 1
    }
  }
}


# **********************************
# Reversed Martingale Wager strategy

reversed_Martingale <- function(params, store){
  for (i in params$series){
    # store money spend for establish position
    if (flagStore[store$iter,i] == 1 |
        flagStore[store$iter,i] == 2){
      revMartingale_lastProfit[i] <<- -moneySpendStore[store$iter+1, i]
    }
```

```r
    # when clear position, calculate pnl
    # when winning, lastProfit = 1 , else 0
    else if(flagStore[store$iter,i] == 5){
      if (revMartingale_lastProfit[i] - moneySpendStore[store$iter+1, i] > 0){
        revMartingale_lastProfit[i] <<- 1
      }else{
        revMartingale_lastProfit[i] <<- 0
      }
    }

    # when clear position and win, double the wager for the next time
    if (flagStore[store$iter,i] == 5 & revMartingale_lastProfit[i] == 1){
      Martingale_multiplier[i] <<- Martingale_multiplier[i] * params$Martingale
    }
    # if win, just reset the wager to the initial amount
    else if (flagStore[store$iter,i] == 5 & revMartingale_lastProfit[i] == 0){
      Martingale_multiplier[i] <<- 1
    }
  }
}

# the Kelly best ratio for each series can only derive from final performance
# using the following function

# **********************************
# performance measurement
perfCalc <- function(test, pfolioPnL){

  lastProfit <- vector(mode="numeric",length=10)
  winTimes <- vector(mode="numeric",length=10)
  WinBalance <- vector(mode="numeric",length=10)
  loseTimes <- vector(mode="numeric",length=10)
  loseBalance <- vector(mode="numeric",length=10)
  tradeTimes <- vector(mode="numeric",length=10)
  gambitRate <- vector(mode="numeric",length=10)
  kelly <- vector(mode="numeric",length=10)
  winRate <- vector(mode="numeric",length=10)
  lossRate <- vector(mode="numeric",length=10)

  pnl <- as.double(pfolioPnL[["pfoliosPnL"]][nrow(pfolioPnL[["pfoliosPnL"]]),2])
  PDratio <- pfolioPnL[["fitAgg"]]
  activeness <- as.double(test[["k"]])

  for (d in 2:1000){
    for (i in 1:10){
      # store money spend for establish position
      if (flagStore[d-1,i] == 1 |
          flagStore[d-1,i] == 2){
        lastProfit[i] <- -moneySpendStore[d, i]
      }
      # when clear position, calculate this trade whether lose or win,
      # when losing, lastProfit = 1 , else 0
      else if(flagStore[d-1,i] == 5){
        if (lastProfit[i] - moneySpendStore[d, i] > 0){
          winTimes[i] <- winTimes[i] + 1
          WinBalance[i] <- WinBalance[i] + (lastProfit[i] - moneySpendStore[d, i])
        }else if (lastProfit[i] - moneySpendStore[d, i] < 0){
          loseTimes[i] <- loseTimes[i] + 1
          loseBalance[i] <- loseBalance[i] + (lastProfit[i] - moneySpendStore[d, i])
        }
        tradeTimes[i] <- tradeTimes[i] + 1
      }
    }
  }

  for (i in 1:10){
    winRate[i] <- winTimes[i]/tradeTimes[i]
    lossRate[i] <- 1-winRate[i]
    gambitRate[i] <- (winTimes[i]/WinBalance[i]) / (loseTimes[i]/loseBalance[i])
    kelly[i] <- ((gambitRate[i]+1)*winRate[i] - 1) / gambitRate[i]
  }

  avg_winRate <- mean(winRate, na.rm = TRUE)
  avg_lossRate <- mean(lossRate, na.rm = TRUE)
  avg_winTimes <- mean(winTimes, na.rm = TRUE)
  avg_loseTimes <- mean(loseTimes, na.rm = TRUE)
  avg_tradeTimes <- mean(tradeTimes, na.rm = TRUE)
  sum_winTimes <- sum(winTimes, na.rm = TRUE)
  sum_loseTimes <- sum(loseTimes, na.rm = TRUE)
  sum_tradeTimes <- sum(tradeTimes, na.rm = TRUE)
```

```
    sum_kelly <- sum(kelly, na.rm = TRUE)
    expected_profit <- pnl / sum_tradeTimes

    res <- list('pnl'=pnl,
                'PDratio'=PDratio,
                'activeness'=activeness,
                'winTimes'=winTimes,
                'loseTimes'=loseTimes,
                'tradeTimes'=tradeTimes,
                'WinBalance'=WinBalance,
                'loseBalance'=loseBalance,
                'winRate'=winRate,
                'lossRate'=lossRate,
                'gambitRate'=gambitRate,
                'kelly'=kelly,
                'avg_winRate'=avg_winRate,
                'avg_lossRate'=avg_lossRate,
                'avg_winTimes'=avg_winTimes,
                'avg_loseTimes'=avg_loseTimes,
                'avg_tradeTimes'=avg_tradeTimes,
                'sum_winTimes'=sum_winTimes,
                'sum_loseTimes'=sum_loseTimes,
                'sum_tradeTimes'=sum_tradeTimes,
                'sum_kelly'=sum_kelly,
                'expected_profit'=expected_profit
                )
    return(res)
  }
```

Like the codes above show, the Kelly best ratio for each series can only derive from calculating the win rate, loss rate, expected gain for each trade using final performance. At this point, we are left with 1 question, how exactly should we apply Kelly formula? Our first idea was to calculate the Kelly best wager ratio separately for different strategies and different series, thus obtaining 40(4*10) wagers. But we found that when there are more parameters, the easier it is to over-fit. In addition, we find it too expensive to develop separate statistics on the performance of each sub-trading strategy in our final Combined strategy. Thus, this idea is abandoned. Finally, we considered the whole trading system as a whole and then used the Kelly formula only for the series, and found that the over-fit was not so serious.

Table 2. Combined Strategy Performance in Training Set (1000 days part 1 + 500 days part 2) with different Wager Strategy

Wager Strategy(initial wager)	PD ratio	PnL	Trigger Times	Invested No. Of Series	Win times	Lose times	Expected Profit per time	Win Rate	Loss Rate
fixed wager ($50,000 worth shares for each series)	-443048.5	-443048.5	293	10	129	163	-1512.11	40.65%	59.35%
fixed wager ($100,000 worth shares for each series)	-260402.3	-260402.3	175	10	82	93	-1488.01	46.75%	53.25%
performance-based fixed wager (based on $50,000 fixed wager PnL, bad series deducted)	-153846.63	-153846.63	164	5	78	86	-938.09	46.60%	53.40%
performance-based fixed wager(based on $100,000 fixed wager PnL, bad series deducted)	2.07	118957.77	29	3	12	17	4101.99	41.11%	58.89%
Fixed Kelly optimal ratio wager	1.05	58645.8	301	7	130	171	194.84	44.23%	55.77%

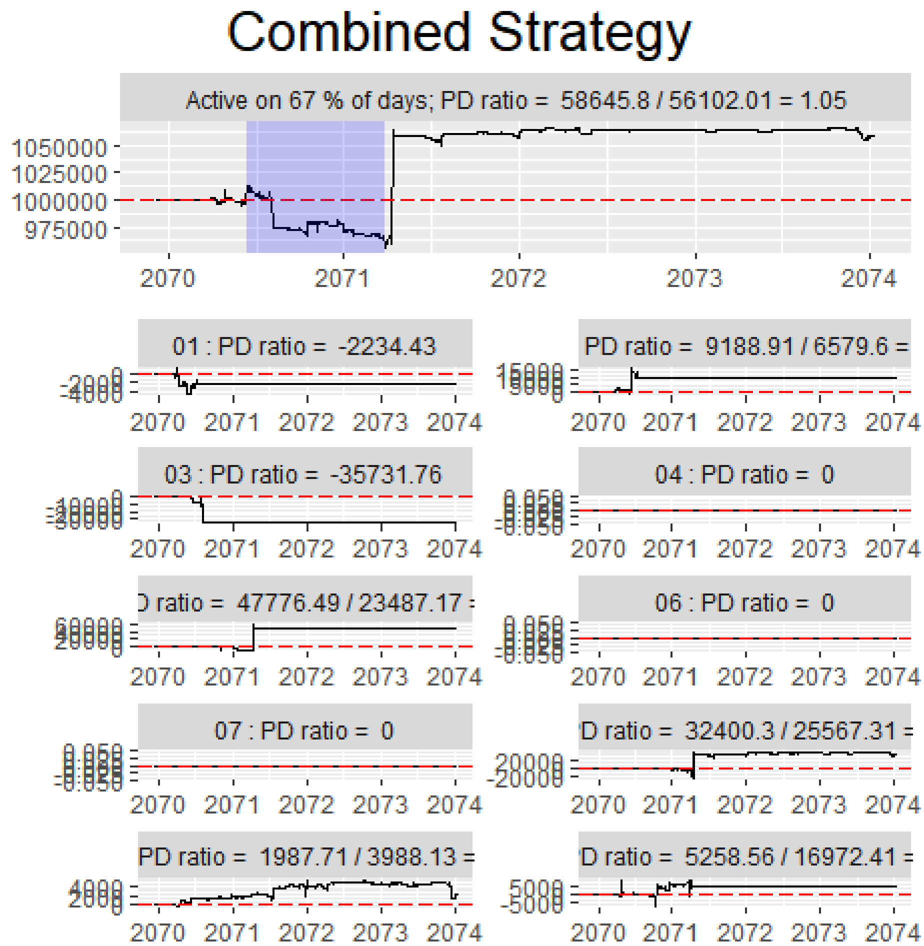
From observing the Table 2 data, it is shown that although the performance-based fixed wager strategy (based on $100,000 fixed wager PnL, bad series deducted) can get 2.07 PD Ratio. However, it is mainly concentrated on only 3 series and the number of trading is very small, which does not meet the requirement of trading frequency at all, and this is also in the case of using the prior knowledge of known performance of fixed wager strategy on the same data set. On the contrary, as can be seen about Kelly's performance, in the first 200 days, it only slightly decreased, with no large drawbacks over time. In the following periods, it gained a relatively good profit. What's more, the number of invested series and trading frequency are in line with the requirements.

The next step is to consider which wager strategy should be selected at series level, Martingale Strategy or Reverse Martingale Strategy. The biggest difference between these two strategies is that one doubles the bet after winning in a series and the other doubles the bet after losing. Just as we test the Kelly formula for viability, the same AB test is launched for these 2 strategies.

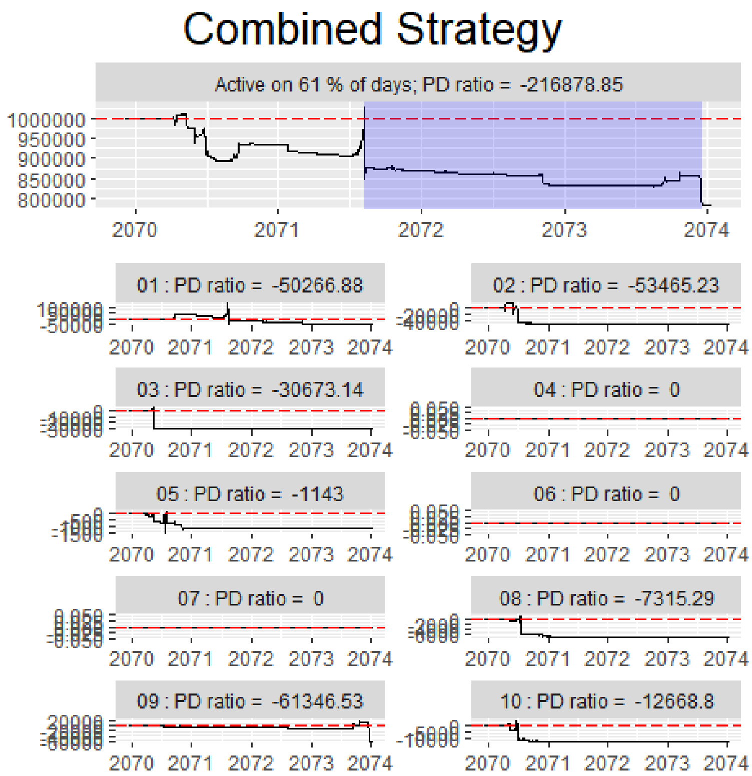
Table 3. Combined Strategy Performance in Training Set (1000 days part 1 + 500 days part2) with Martingale Strategy and Reverse Martingale Strategy

Wager Strategy(technical level)	PD ratio	PnL	Trigger Times	Win times	Lose times	Expected Profit per time	Win Rate	Loss Rate
Martingale	1.05	58645.8	301	130	171	194.84	44.23%	55.77%
Reverse Martingale	-216878.85	-216878.85	297	134	163	-730.23	45.29%	54.71%
Control Group(without these 2 strategies)	-41472.99	-41472.99	476	212	264	-87.13	41.40%	58.60%

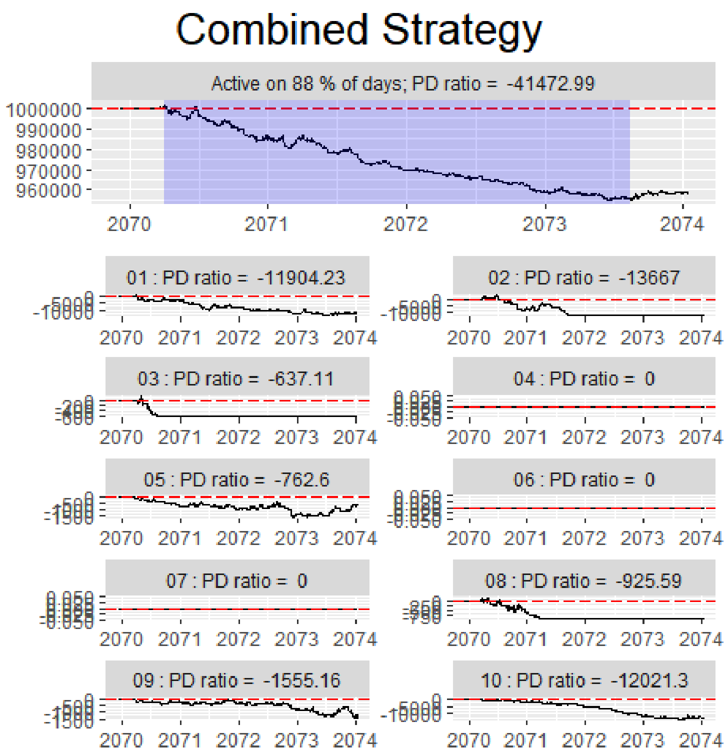
Graph 5. Performance of Combined Strategy with optimized parameters and Martingale in Training Set (1000 days part 1 + 500 days part2)



Graph 6. Performance of Combined Strategy with optimized parameters and Reverse Martingale in Training Set (1000 days part 1 + 500 days part2)



Graph 7. Performance of Combined Strategy with optimized parameters and without wager strategy in Training Set (1000 days part 1 + 500 days part2)



According to Table 3, the Martingale Strategy obviously outperforms the other 2 strategies, except that win rate for Martingale is about 1% lower than the Reverse Martingale strategy and 175 fewer Trigger Times than the control group. In addition, the Reverse Martingale Strategy and the control group were consistently losing money on any series during the experimental period.

Regarding the exploration of capital utilization and leverage, our initial idea was to get many performance matrices, such as total capital utilization, average capital utilization and leverage, by increasing or decreasing positions proportionally by 10 series. Then by using these indicators, we can get the relationship between profit and loss indicators and capital utilization and leverage, so as to find the optimal capital utilization and leverage. However, we gave up the development of this idea because of the code complexity and lack of time.

Apart from position sizing, like what mentioned in Section 1, the division of the data set is also very important. As suggested in the course, the dataset can be divided into part 1 data as training set and part 2 as test set. But the problem is that the transaction data is discrete time series data and highly path dependent, but the features may be completely inconsistent at relatively long-time intervals. Therefore, it is very difficult to find a reasonable length of the segmented data set. We have tried to train our parameters by defining the training set as 1000, 1200, 1500, 1800 days, but finally found that only the split of 1500/500 days works best for 10 series. If the training set takes too short, underfitting may occur, resulting in strategy that ends up performing poorly. If the training set takes too long, it is easy to over-fit the training data to get good performance only in the training set and poor performance in the test or validation set. We decided to use 1500 days/500 days splitting method to split the part 1 and part 2 data sets by observing the performance using different strategies and parameters, which were very stable over time.

2.3 Considering the strategy compare with alternatives.

Because even if an effective strategy can be found based on the historical data, there is no way to find out how long that strategy will continue to be effective in the future, and how reliable it can be. Thus, integrated learning is a more reliable system because a strategy is certainly easier to fail and more fragile than an integrated trading system.

Inspired by the concept of integration learning in the field of machine learning, at the beginning 3 natural ideas about trading system design came up to us.

Boosting, we ask the opinions of the four strategies daily based on their performance on training set, from the best-performing strategy to the worst-performing strategy, and then take the opinion of the better performed strategy that was the first to decide to place an order.

Bagging, each day we poll 4 strategies, and then we get their judgment on each series, and finally take the plurality of the judgment execution.

Stacking, we allocate different amounts of money to each of the four strategies based on their performance on training set, and then sum the number of orders placed every day in the following. Due to the high development cost, we cannot develop all 3 systems and then launch AB experiments to find the optimal system. There is a famous saying in machine learning: No free lunch, which means that no matter what model or strategy is used, the theoretical result should be the same under the same scenario and the same batch of data. In addition, we developed the Boosting system first, therefore we ended up using this trading system directly instead of trying the other 2.

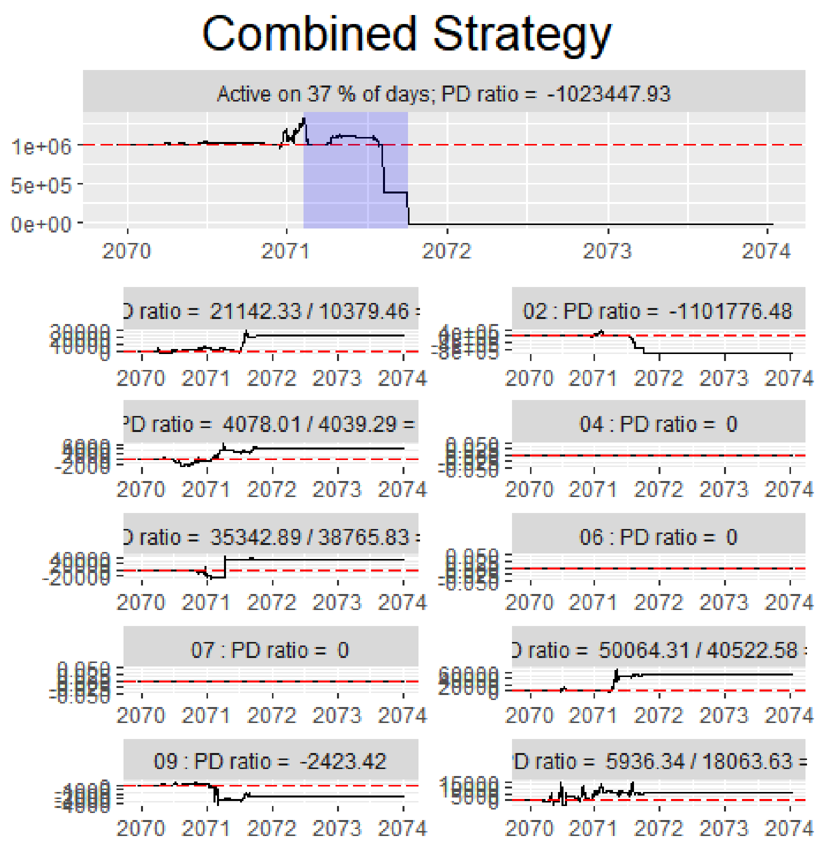
2.4 Managing risk in the strategy

As explained in Section 1, there are 5 risk strategies: 1 take-profit strategy, 3 stop-loss strategies and 1 strategy exit mechanism. To explore whether these risk strategies are effective and how these risk management strategies affect our trading system, similarly, a series of AB experiments was launched.

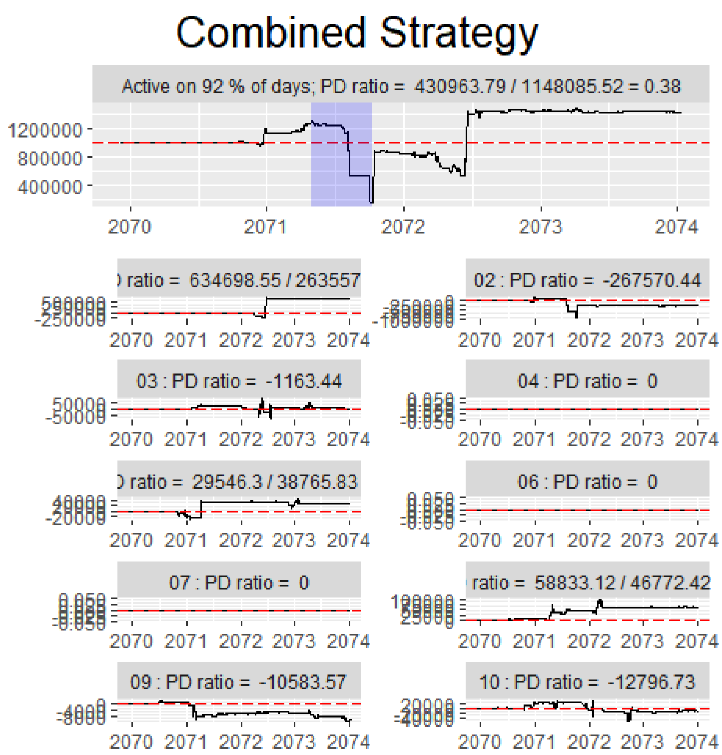
Table 4. Combined Strategy Performance in Training Set (1000 days part 1 + 500 days part2) with different Risk Management Strategy

Risk Management Strategy	PD ratio	PnL	Trigger Times	Win times	Lose times	Expected Profit per time	Win Rate	Loss Rate
Control Group(no risk strategy)	-1023447.93	-1023447.93	146	48	98	-6764.63	33.19%	66.81%
Drawdown Stopgain	0.38	430963.79	332	139	193	1298.08	41.15%	58.85%
Drawdown Stop-gain + Time Stops	-1317377.46	-1317377.46	478	219	259	-1324.42	45.79%	54.21%
Drawdown Stop-gain + Time Stops + Trailing Stops	-1071837.34	-1071837.34	395	185	210	-1795.94	46.69%	53.31%
Drawdown Stop-gain + Time Stops + Trailing Stops + VaR_ES_Stops	-1071837.34	-1071837.34	395	185	210	-1795.94	46.69%	53.31%
Drawdown Stop-gain + Time Stops + Trailing Stops + VaR_ES Stops + Strategy Exit	1.05	58645.8	301	130	171	194.84	44.23%	55.77%

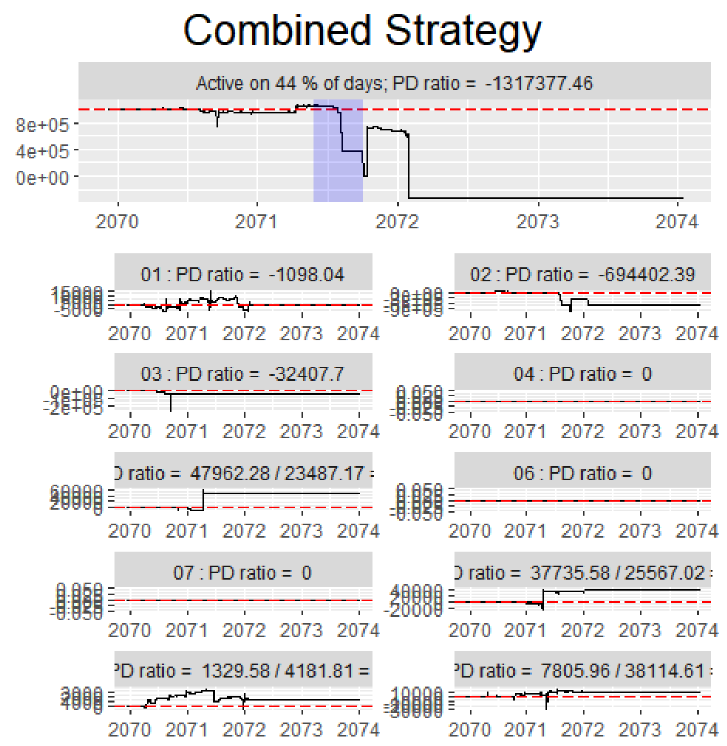
Graph 8. Performance of Combined Strategy with optimized parameters and without risk management in Training Set (1000 days part 1 + 500 days part2)



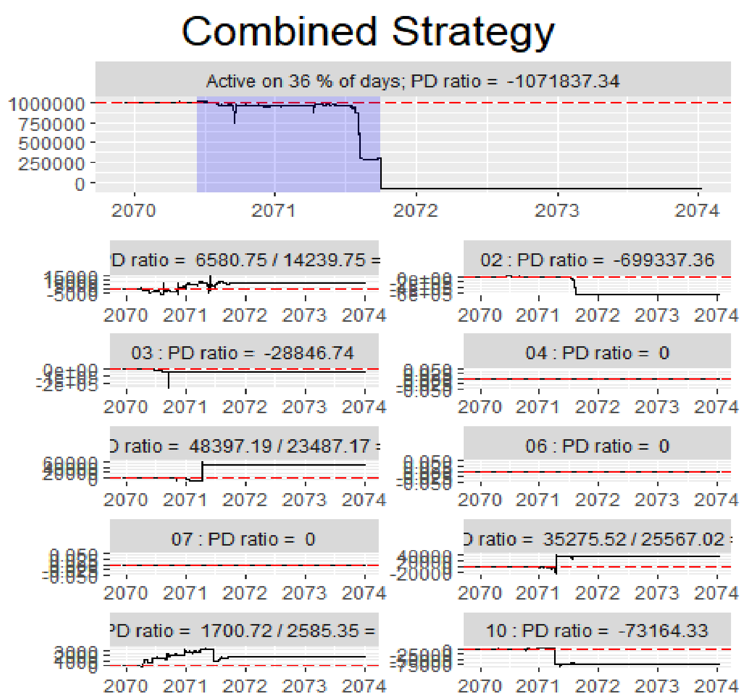
Graph 9. Performance of Combined Strategy with final parameters and with Drawdown Stop-gain in Training Set (1000 days part 1 + 500 days part2)



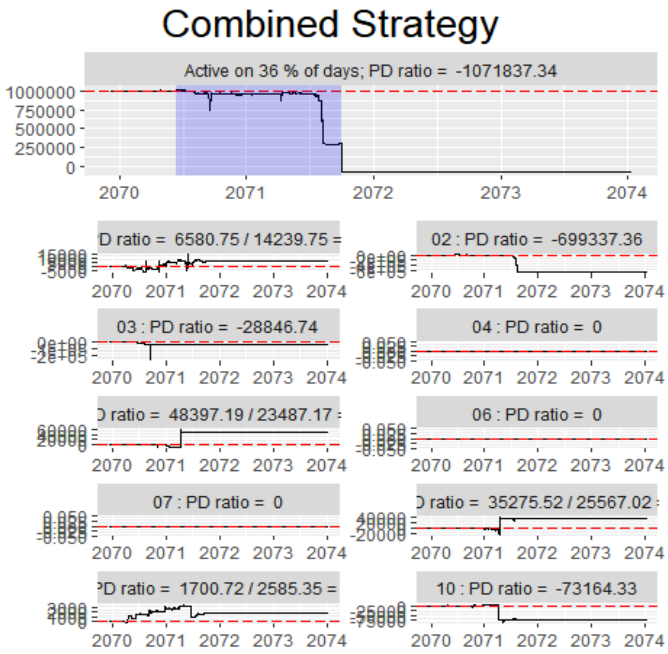
Graph 10. Performance of Combined Strategy with final parameters and with Drawdown Stop-gain + Time Stops in Training Set (1000 days part 1 + 500 days part2)



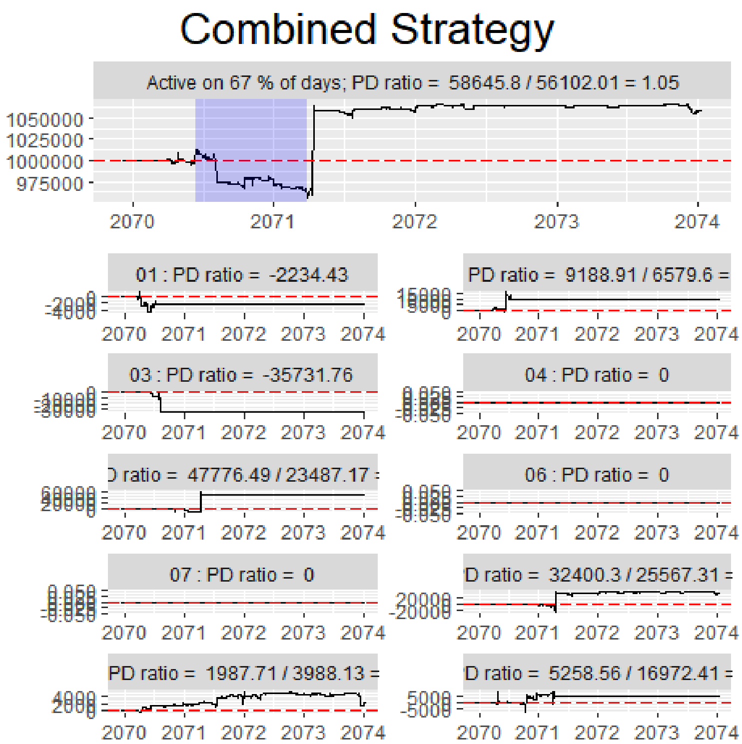
Graph 11. Performance of Combined Strategy with final parameters and with Drawdown Stop-gain + Time Stops + Trailing Stops in Training Set (1000 days part 1 + 500 days part2)



Graph 12. Performance of Combined Strategy with final parameters and with Drawdown Stop-gain + Time Stops + Trailing Stops + VaR_ES_Stops in Training Set (1000 days part 1 + 500 days part2)



Graph 13. Performance of Combined Strategy with final parameters and with Drawdown Stop-gain + Time Stops + Trailing Stops + VaR_ES Stops + Strategy Exit in Training Set (1000 days part 1 + 500 days part2)



As you can see from the table and graphs above, the biggest impact on our trading system is on Drawdown Stopgain and Strategy Exit. Comparing the data of Control Group and Drawdown Stopgain, as can be observed that all indicators of performance have a very strong positive impact by Stopgain method. After adding 3 stop-loss strategies, although the total number of Trigger Times and Win Rate increased, the losses were huge on a few series, leading to the system going bankrupt eventually. This shows that although our stop-loss strategies were able to prevent the loss from expanding on some series, on a few series it led to reopening the position quickly due to the unreasonable setting of the time interval between trades especially making the loss bigger when combined with the Martingale strategy. And more frequent large amount of entering led us to suffer more slippage. In some series, such as series 1, due to unreasonable parameter settings in the risk strategy, premature exit of positions led to a minor loss directly from a large profit. This phenomenon was improved after the strategy exit mechanism was finally added.

3.Evaluation and analysis of performance on part 3

3.1 The strategy performs relative to expectations

3.1.1 Dataset Breakdown

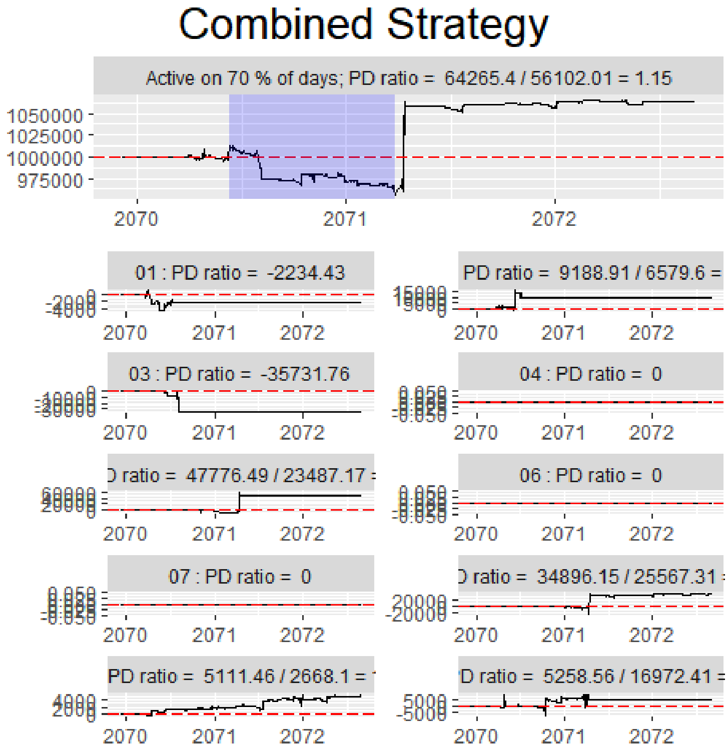
From Table 5, we can see that the Activeness in part 3 compared with part 1 decreased by 40%, and the number of Trigger Times decreased by 26%, while all other matrix largely increased, indicating that our trading system are quite stable in part 1 and part 3, and our trading system has learned the pattern of data in part 1 quite well. However, the trading signal parameters in the trading strategy seem over-fit, which leads to a lower Trigger Times. The trading signal parameters can be adjusted subsequently to increase the Trigger Times.

In addition, by comparing the data of part 2 and part 3, we can learn that except for the win rate of part 3, which is 2% higher than part 2, all other indicators have dropped significantly, which shows that our trading system obviously over-fit on the part 2 data, but this is within expectations. Because our training set includes the first 500 days of data of part 2, which is future information that is not supposed to be known. Therefore, the performance on the part 2 data is not confident.

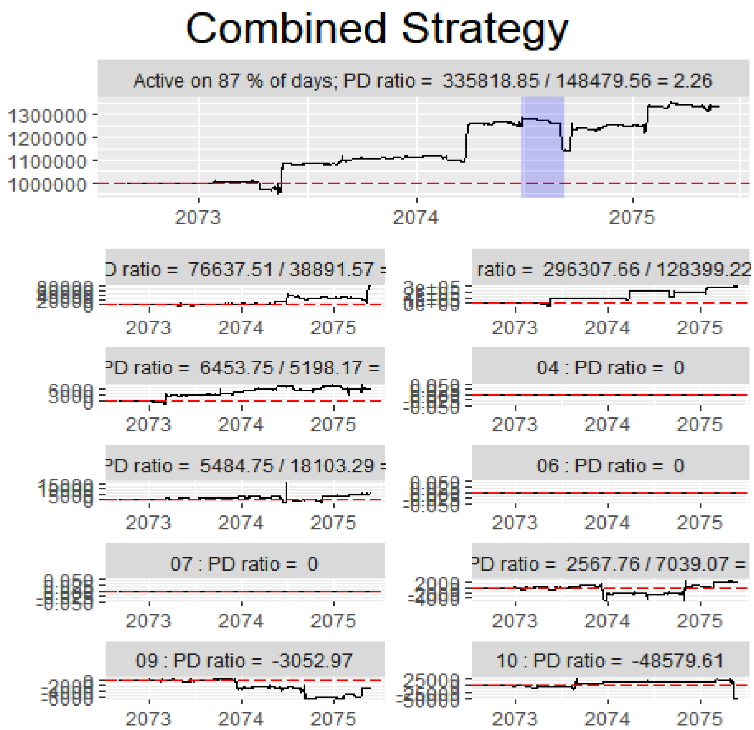
Table 5. Performance of final submitted Strategy in 3 datasets

Dataset	PD ratio	PnL	Max Drawdown	Activeness	Trigger Times	Win times	Lose times	Expected Profit per time	Win Rate	Loss Rate
Part 1	1.15	64265.4	56102.01	70%	301	129	172	213.51	44.08%	55.92%
Part 1 vs. Part 3	3.48%	32.05%	26.99%	-40.00%	-26.58%	-10.85%	-38.37%	79.85%	13.97%	-11.02%
Part 2	2.26	335818.85	148479.56	87.00%	650	321	328	516.64	49.18%	50.82%
Part 2 vs. Part 3	-47.35%	-74.73%	-52.02%	-51.72%	-66.00%	-64.17%	-67.68%	-25.68%	2.16%	-2.09%
Part 3	1.19	84860.68	71241.39	42%	221	115	106	383.99	50.24%	49.76%

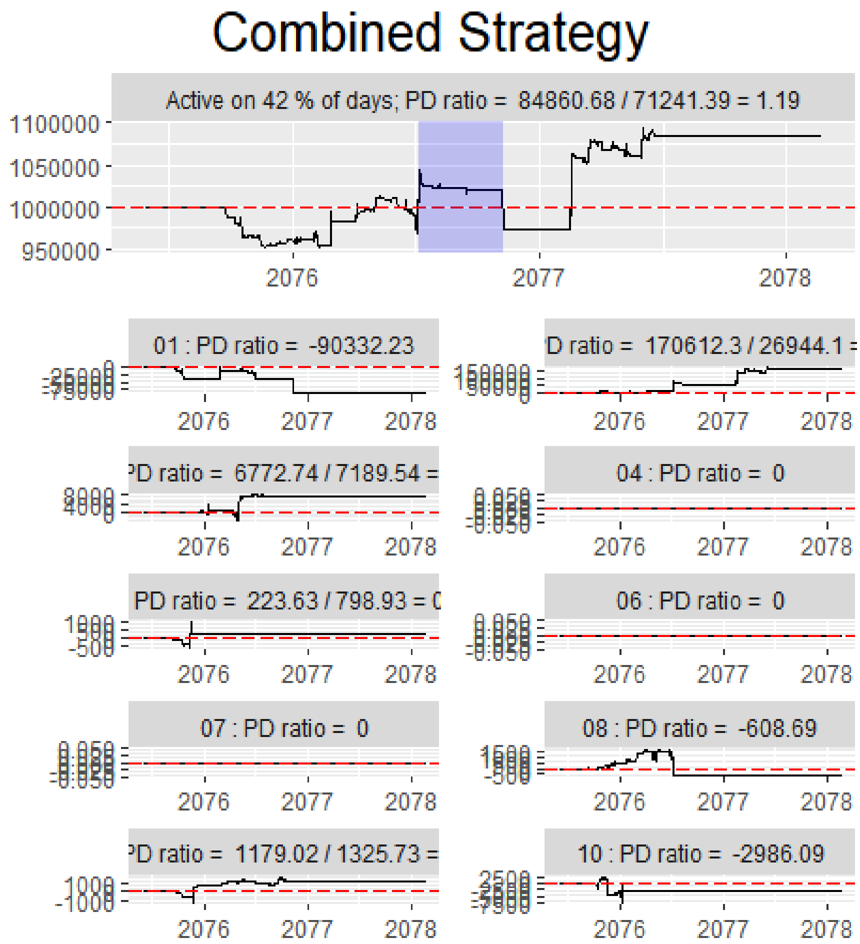
Graph 14. Performance of final submitted Strategy in Part 1



Graph 15. Performance of final submitted Strategy in Part 2



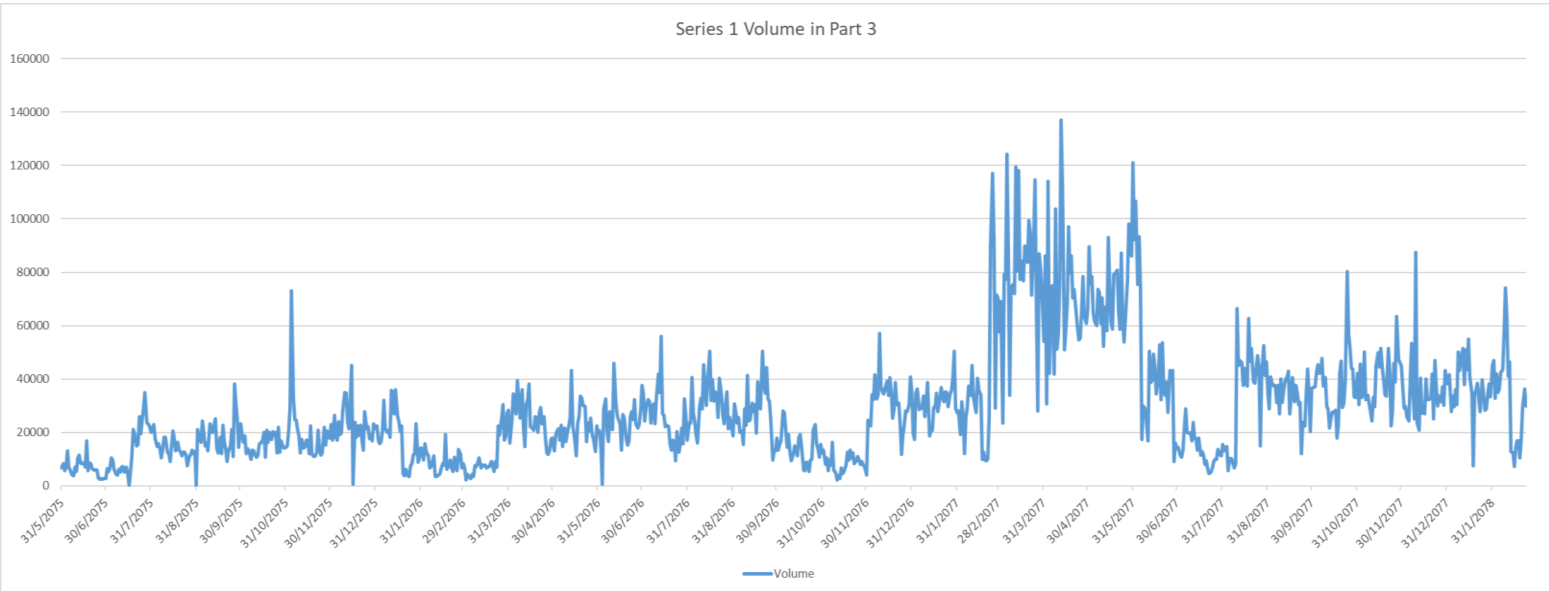
Graph 16. Performance of final submitted Strategy in Part 3



Comparing part 1 and part 3 data, we can see that series 1 is the greatest loss series in terms of overall trading system losses, and the losses on series 1 had two major great drawbacks, which occurred near June and October 2076. According to Graph 17, we can see that OHLC has been in a steady upward trend around June and October 2076, and no significant abnormal data fluctuations have observed. Based on Graph 18, as can be seen, Volume in series 1 is in a fluctuating upward trend in June and a fluctuating downward trend in October 2076, respectively.

Graph 17. Series 1 OHLC price in Part 3

Graph 18. Series 1 Volume in Part 3



After ruling out the data anomalies, we are basically sure that the loss on series 1 in part 3 data is due to our trading system itself.

3.1.2 Series Breakdown

According to Table 6, we can see that in part 1, our strategy performs poorly on series 3, 5, 8, 9. Win Rate on series 3, 5, 8 are less than 40%, while the odds-on series 3 and 9 are the lowest, at -35.11 and -3.84 respectively. But as marked in Table 7, the performance of our trading system on different series in part 3 has completely changed, with the performance on series 1, 2, 10 being the worst. Also, as can be noted, the Kelly's optimal bet ratio on all 7 series has changed greatly.

Table 6. Series Breakdown Performance of final submitted Strategy in Part 1

Series	Win Balance	Loss Balance	Gambit Rate	Kelly	Trigger Times	Win times	Lose times	Win Rate	Loss Rate
1	5839.61	-7144.13	-1.43	0.86	13	7	6	53.85%	46.15%
2	193508.55	-118431.11	-1.10	0.97	14	9	5	64.28%	35.71%
3	342.50	-36074.26	-35.11	0.27	12	3	9	25%	75%
4	0	0	0	0	0	0	0	0	0
5	84221.38	-10379.86	-0.06	12.58	32	10	22	31.25%	68.75%
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	62356.47	-38829.28	-0.41	1.88	96	38	58	39.58%	60.42%
9	10289.88	-49143.22	-3.84	0.59	92	41	51	44.57%	55.43%
10	76028.21	-41036.17	-0.54	1.43	42	21	21	50%	50%

Table 7. Series Breakdown Performance of final submitted Strategy in Part 3

Series	Win Balance	Loss Balance	Gambit Rate	Kelly	Trigger Times	Win times	Lose times	Win Rate	Loss Rate
1	31434.62	-121766.85	-2.26	0.65	19	7	12	36.8%	63.2%
2	315370.41	-1059259.31	-3.14	0.65	60	29	31	48.33%	51.67%
3	94530.02	-48315.12	-0.51	1.48	36	18	18	50%	50%
4	0	0	0	0	0	0	0	0	0
5	95159.95	-454.37	-0.01	67.89	7	4	3	57.14%	42.86%
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	582451.01	-427564.72	-0.90	1.05	40	22	18	55%	45%
9	275057.77	-1178.12	-0.01	21.39	39	29	10	74.36%	25.64%
10	442807.33	-1179066.94	-1.14	0.91	20	6	14	30%	70%

3.1.3 Trading Breakdown

As is shown in Table 8 and Table 9, we found that Sum Long Enter Times + Sum Short Enter Times is not equal to Sum Exit Times, which means there is a bug in our trading signal recording system. For instance, in Table 8 Series 1, Long Enter Times + Sum Short Enter Times is greater than Sum Exit Times, which means that the number of exit times is more than the number of enter times, which is obviously not in line with reality. Secondly, according to the Trigger Times of 4 trading strategies on the 10 series, we can find that most trades are mainly concentrated in the TMA Strategy, the top priority BBTF strategy only triggered a few times, while the last 2 strategies did not trigger at all. This may be due to the bug in the signal system, or the trading signal limits for BBTF, BBMR and LMV strategies are too strict, or the trading signal limits for TMA strategy are far too loose, which squeezes the trigger opportunities of the other strategies.

Table 8. Trading Breakdown Performance of final submitted Strategy in Part 1

Series	Sum Long Enter Times	Sum Short Enter Times	Sum Long Hold Times	Sum Short Hold Times	Sum Exit Times	BBTF Trigger Times	TMA Trigger Times	BBMR Trigger Times	LMV Trigger Times
1	6	6	23	25	13	3	70	0	0
2	5	5	15	23	14	9	53	0	0
3	5	11	0	36	12	2	62	0	0
4	0	0	0	0	0	0	0	0	0
5	43	8	85	21	32	18	171	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	69	44	199	151	96	24	535	0	0
9	46	42	243	147	92	16	554	0	0
10	8	24	21	112	42	9	198	0	0

Table 9. Trading Breakdown Performance of final submitted Strategy in Part 3

Series	Sum Long Enter Times	Sum Short Enter Times	Sum Long Hold Times	Sum Short Hold Times	Sum Exit Times	BBTF Trigger Times	TMA Trigger Times	BBMR Trigger Times	LMV Trigger Times
1	106	11	33	31	19	14	186	0	0
2	50	14	158	7	60	17	272	0	0
3	18	17	67	48	36	2	184	0	0
4	0	0	0	0	0	0	0	0	0
5	1	3	3	12	7	0	26	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	11	10	46	83	40	3	187	0	0
9	15	22	51	86	39	4	209	0	0
10	11	1	41	2	20	2	73	0	0

3.2 The mistakes at the technical level, planning and teamwork

3.2.1 Strategy mistakes

1. Lack of market filter

In the part 1 data analysis we found a large number of data anomalies in the series, such as sudden spikes or sudden drops in trading volume, but we did not make good use of this information in the final submitted trading system.

2. Bad feature engineering

Four strategies are basically single-trading factor strategies, which do not exploit factor characteristics, interactions between factors, and timing characteristics well.

3. Fixed Kelly Optimal Ratio

Due to the code complexity limitation, Kelly optimal position ratio is set to fixed, but by comparing the performance of the final submitted trading system on part 1 and part 3 data, we can know that the 10 series are changing over time, and the win rate, loss rate and odds change dramatically, resulting in the initial optimal position ratio is no longer applicable.

4. The strategy exit mechanism did not consider the sub-strategy dimension, but treats the whole trading system as one.

5. Limit order and Market Making Strategy are not utilized

Due to the code complexity limitation of the trading signal recording system, the limit order and market making strategy are not utilized. While the limit order can significantly reduce our slippage error, and the MM strategy is known to perform quite well on both part 1 and part 2 data.

6. The lack of position-adding strategy

Assume that the trading system or strategy in a few series perform continuous good in the short period of time, and there is no a judgment to add positions to expand the profitability.

7. Strategy Exit Mechanism is too simple

The trading system exits all positions when triggers risk management strategy or exit conditions in 4 trading strategies, without a mechanism to gradually reduce positions, resulting in the possibility of frequent trading in the short-term relatively high volatile market.

8. There may be bugs or bad design in the trading signaling system

From previous analysis, we know that in part 3, the trading is mainly concentrated in the TMA strategy, while the other 3 strategies have basically not been triggered. The trading system may have a mechanism design problem or coding bug. For example, the TMA condition is too easy to trigger. This problem should be tested in the part 2 design stage.

3.2.2 Coding mistakes

1. Instead of using intermediate variables to reduce computation, global variables are used frequently, which leads to low fault tolerance, and is not conducive to subsequent iterations.

2. Trading signal recording system design problems

Trading signaling system used "today's position + today's market order" instead of each series of the daily position changes to get, resulting in the signaling logic of the limit order signal being too complex by using money spent calculation module, and finally gave up the attempt. What's worse, the trading signal recording system may have a bug.

3.2.3 Planning and teamwork mistakes

1. Insufficient communication between team members.

During the implementation phase, because each person was responsible for a difficult part of the code, a lot of time was spent on developing the code, which reduced the time for communication, and the communication between group members was sometimes not clear to their own group members due to some over-extensive knowledge.

2. Inefficient learning.

In the first phase we came up with a lot of ideas and presented them in our presentation, but in the development phase we underestimated the development time of the part of the code we were responsible for, which resulted in not fully implementing each of the ideas.

3. No team leader.

As we all became more familiar with the project this year, we did not have a team leader, but it turned out that the importance of the team leader was not only to divide the work among the team members, but also to lead the team to follow the planned schedule.

3.3 What have we learnt and how would we do things differently

3.3.1 Strategy and Coding Improvement

First of all, a market filter should be added, such as making corresponding operational strategies according to the quantitative determination of whether the data (OHLCV, trading indicators) is abnormal according to the 3-sigma principle, and the performance of each strategy during the period of abnormal data on the training set. We can stop trading particular series if a series is judged to be abnormal. In addition, the market filter should quantitatively define the bull and bear markets, and then the performance of each of the four strategies in the bull and bear markets can be derived on the training set, so that different trading strategies can be activated in the corresponding market conditions to obtain better performance.

In addition, we should strengthen the feature engineering. For example, we can activate different system designs (boosting, bagging, stacking) in different periods after the failure of particular system, mix multiple signals or use multiple factors at the same time to filter noise, combine multiple factors together.

Periodically, the initial position for each trade for each series should be adjusted according to the Kelly optimal position. For instance, every 100 days recalculate the performance of the trading system in 10 series over the last 100 days, and if the overall Kelly changes more than 10% than before, the initial position will be readjusted to the new Kelly value.

The strategy exit mechanism should calculate the number of continuous loss times of each trading strategy on each series, and if a strategy reaches the customized parameters on a certain series, then exit this strategy for all subsequent trades on this series instead of prohibiting this series from all trading strategies.

The trading signaling system should be redesigned based on daily position changes instead of "today's position + today's market order", and include limit order and MM strategies.

A position adding/decrease strategy to gradually increase or decrease positions according to data analysis. Say if a strategy has a win rate above 70% in the last 50 days on a series, and we already held certain positions in this series and the enter signal is triggered again, then we can increase our position in this series again.

3.3.2 Planning and Teamwork Improvement

1. More precise division of work for team members.

For example, those who are good at coding can do more code development, those who are good at writing can do more report writing, and those who are good at leading and coordinating information within the team can become team leader.

2. Improve the ability of team members to execute learning and learn effectively.

In this project, we searched for a lot of ideas in the design phase, but in the second phase, some of the ideas were not developed because of the lack of ability to learn new knowledge, which is a great pity. In addition, the actual time taken to develop the genetic algorithm was much longer than we had initially anticipated. With regard to learning effectiveness, the lack of sufficient discernment when searching for relevant information led to a lot of time being wasted on sifting through invalid resources during subsequent discussions

3. Improving communication skills between team members.

The team's efficiency is greatly enhanced by good understanding and communication, especially when communicating with team members about the code they are responsible for developing.

4. Improve team members' thinking skills.

Trying to grasp the essence of a problem and solving large and complex problems often requires structured thinking skills - breaking down a vague problem into something that can be analyzed in detail is the first step in thinking and solving problems.

4. Breakdown of team work

Chia-Wei Liu

Assignment 2:

1. Co-designing and developing trading system
2. Searching for information about the implementation of Martingale
3. Working with team members to integrate all information
4. Completing the development of the trading strategy
5. Developing code of Martingale
6. Integrating trading strategies
7. Developing code for risk control section

Assignment 3:

1. Reviewing with team members the shortcomings of the strategy, such as over-fitting and insufficient optimal robustness tests.
2. Reviewing with the team members the possible new ideas that can be developed if we continue to deepen the process, such as market condition filtering.
3. Writing section 1 Final choice of submitted strategy

Shuying Zhai

Assignment 2:

1. Co-designing and developing trading system.
2. Accessing information on the implementation of Kelly formula.
3. Working with team members to integrate all information.
4. Completing the development of the trading strategy.
5. Developing code of Kelly formula section.
6. Adding robustness checks following feedback.

Assignment 3:

1. Reviewing with team members the shortcomings of the strategy, such as over-fitting and less than optimal robustness tests.
2. Reviewing with the team members the possible new ideas that can be developed if we continue to deepen the process, such as market condition filtering.
3. Writing section 2 Justification of submitted strategy.

Yushan Zhang

Assignment 2:

1. Co-designing and developing trading system.
2. Accessing relevant information on the implementation of genetic algorithms.
3. Working with team members to integrate all information.
4. Developing of code related to genetic algorithms section.
5. Completing parameter optimization.
6. Adjusting trading signals based on parameter optimization results.

Assignment 3:

1. Reviewing with team members the shortcomings of the strategy, such as over-fitting and less than optimal robustness tests.
2. Reviewing with the team members the possible new ideas that can be developed if we continue to deepen the process, such as market condition filtering.
3. Writing section 3 Evaluation and analysis of performance in part 3.