For Loop code for stocks data analysis

Aayush

2024-06-24

```
library(pacman)
```

```
## Warning: package ’pacman’ was built under R version 4.3.2
```

```
pacman::p_load(tidyverse)
```

```
# Install and load the readxl package
library(readxl)

# Specify the path to your Excel file
file_path <- "C:\\Users\\Aayush\\Documents\\dr. moore stock project\\excel files\\Liquidity.xlsx"

# List all sheet names in the Excel file
sheet_names <- excel_sheets(file_path)

# Read all sheets into a list of data frames
data <- lapply(sheet_names, function(sheet) {
  read_excel(file_path, sheet = sheet, col_names = FALSE)
})
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## * ‘‘ -> ‘...1‘
## * ‘‘ -> ‘...2‘
## * ‘‘ -> ‘...3‘
## * ‘‘ -> ‘...4‘
## * ‘‘ -> ‘...5‘
## * ‘‘ -> ‘...6‘
## * ‘‘ -> ‘...7‘
## * ‘‘ -> ‘...8‘
## * ‘‘ -> ‘...9‘
## * ‘‘ -> ‘...10‘
## * ‘‘ -> ‘...11‘
## * ‘‘ -> ‘...12‘
## * ‘‘ -> ‘...13‘
## * ‘‘ -> ‘...14‘
## * ‘‘ -> ‘...15‘
## * ‘‘ -> ‘...16‘
## * ‘‘ -> ‘...17‘
## * ‘‘ -> ‘...18‘
## * ‘‘ -> ‘...19‘
## * ‘‘ -> ‘...20‘
## * ‘‘ -> ‘...21‘
## * ‘‘ -> ‘...22‘
## * ‘‘ -> ‘...23‘
## * ‘‘ -> ‘...24‘
## * ‘‘ -> ‘...25‘
```

```
# Optionally, name each element of the list with the corresponding sheet name
names(data) <- sheet_names

# View the list of data frames
head(data)
```

```
## $High
## # A tibble: 100 x 25
##     ...1  ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9 ...10 ...11 ...12 ...13
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  73.7 12.3   34.6  88.3  46.8  55.5  27.5  77.7  37.0  47.0  86.9  48.4  20.6
##  2  74.9 13.0   35.7  86.4  49.3  55.0  27.6  82.4  35.4  49.6  90.9  50.7  21.7
##  3  72.6 11.0   36.2  83.2  49.7  53.6  28.3  83.8  36.8  48.4  92.3  52.8  20.4
##  4  74.8 12.0   37.4  87.3  51.2  51.4  26.8  81.2  35.7  47.6  94.0  53.8  21.7
##  5  72.5 12.8   38.2  89.5  52.6  54.1  26.1  81.6  35.9  46.5  91.6  54.5  21.8
##  6  70.6 11.8   37.8  90.5  52.6  55.9  27.1  82.1  35.0  49.3  93.6  53.0  23.4
##  7  68.8 12.1   36.6  87.6  50.1  56.7  25.6  84.3  35.5  51.3  92.0  51.4  22.0
##  8  72.1 10.6   36.1  87.7  50.7  56.8  26.0  85.3  37.1  49.1  90.7  52.3  22.9
##  9  74.0 11.2   36.8  90.8  53.0  60.1  26.2  87.9  37.5  47.8  89.0  54.1  22.3
## 10  77.6  9.93  37.3  93.5  51.3  61.4  27.4  87.6  39.6  48.4  87.7  53.4  23.0
## # i 90 more rows
## # i 12 more variables: ...14 <dbl>, ...15 <dbl>, ...16 <dbl>, ...17 <dbl>,
## #   ...18 <dbl>, ...19 <dbl>, ...20 <dbl>, ...21 <dbl>, ...22 <dbl>,
## #   ...23 <dbl>, ...24 <dbl>, ...25 <dbl>
##
## $Low
## # A tibble: 100 x 25
##     ...1  ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9 ...10 ...11 ...12 ...13
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  73.4  9.56  31.7  86.0  44.6  53.3  25.9  76.7  33.1  46.0  84.7  45.5  17.6
##  2  71.1  9.80  32.8  84.2  45.4  54.5  24.6  79.0  32.1  45.6  87.7  48.1  16.8
##  3  70.1 10.1   33.3  80.4  48.1  51.3  25.3  81.1  34.1  46.4  90.3  49.9  19.3
##  4  73.3 10.2   34.3  84.5  49.1  48.0  23.8  79.5  32.3  43.9  91.6  49.6  17.8
##  5  68.8 10.6   35.0  86.9  49.2  51.7  23.4  77.8  32.6  43.6  88.9  51.9  17.4
##  6  70.0  8.71  35.1  88.4  49.3  52.0  24.1  79.1  33.6  46.7  90.9  49.2  20.1
##  7  65.9  9.84  35.0  84.1  48.8  53.3  23.3  82.0  34.8  47.9  88.8  50.3  19.5
##  8  68.4  7.53  33.1  86.0  46.8  56.1  25.0  83.8  35.7  47.2  89.2  48.8  20.1
##  9  71.6  8.29  33.8  88.6  50.1  57.5  24.6  85.2  35.8  44.6  86.2  50.9  19.9
## 10  74.0  8.04  34.7  91.4  49.5  59.2  23.6  85.1  37.4  45.2  83.6  52.0  20.3
## # i 90 more rows
## # i 12 more variables: ...14 <dbl>, ...15 <dbl>, ...16 <dbl>, ...17 <dbl>,
## #   ...18 <dbl>, ...19 <dbl>, ...20 <dbl>, ...21 <dbl>, ...22 <dbl>,
## #   ...23 <dbl>, ...24 <dbl>, ...25 <dbl>
##
## $Open
## # A tibble: 100 x 25
##     ...1  ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9 ...10 ...11 ...12 ...13
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  73.6 11.1   33.4  86.6  45.9  53.4  26.9  77.5  35.3  46.8  85.0  47.5  18.1
##  2  73.1 11.4   35.5  86.3  47.8  54.8  25.6  81.4  34.0  47.6  89.3  50.0  19.7
##  3  71.0 10.1   35.1  82.7  49.6  52.5  25.6  83.1  35.4  47.1  91.2  52.2  20.0
##  4  74.5 11.0   36.2  86.5  50.1  49.9  25.2  80.8  34.2  45.2  93.8  52.1  20.7
##  5  70.3 11.0   36.3  88.4  50.9  52.8  25.5  79.5  34.1  45.3  90.7  53.3  19.3
##  6  70.3  9.30  36.0  90.5  50.7  53.9  25.4  80.7  34.3  47.3  91.5  50.8  21.6
##  7  68.3 10.4   35.2  86.2  49.0  54.8  25.1  83.5  35.1  49.7  90.0  50.8  21.2
##  8  70.7  8.90  34.3  86.7  48.3  56.7  25.5  85.1  36.4  48.4  90.0  50.4  21.8
##  9  72.0  9.09  36.1  90.1  51.2  58.6  25.1  86.8  37.2  46.1  88.3  52.7  21.3
## 10  75.7  8.70  35.4  93.2  50.7  60.6  26.1  85.8  37.6  46.7  85.4  53.2  21.9
## # i 90 more rows
## # i 12 more variables: ...14 <dbl>, ...15 <dbl>, ...16 <dbl>, ...17 <dbl>,
## #   ...18 <dbl>, ...19 <dbl>, ...20 <dbl>, ...21 <dbl>, ...22 <dbl>,
## #   ...23 <dbl>, ...24 <dbl>, ...25 <dbl>
##
## $Close
## # A tibble: 100 x 25
##     ...1  ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9 ...10 ...11 ...12 ...13
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  73.5 11.7   33.3  86.6  45.5  53.8  26.6  77.1  34.7  46.4  85.7  48.3  18.8
##  2  73.0 11.1   34.5  85.6  46.9  54.8  26.2  80.5  34.3  47.6  88.9  50.1  18.7
##  3  71.9 10.9   35.0  81.8  48.7  52.6  26.4  82.6  35.6  46.6  91.5  51.4  19.6
##  4  74.2 11.0   35.5  85.7  49.6  50.2  25.7  81.1  33.9  46.2  93.5  51.4  19.7
##  5  70.8 10.7   35.9  87.5  50.7  52.2  25.0  80.3  33.5  45.7  89.8  52.6  20.2
##  6  70.2 10.2   35.4  89.5  50.3  54.4  25.1  80.5  34.1  47.7  91.9  51.1  21.1
##  7  67.4 10.2   35.0  85.5  49.9  54.2  25.1  84.2  35.4  49.1  90.6  51.2  21.5
##  8  70.2  9.74  34.3  86.2  49.1  56.5  25.3  84.7  36.4  49.0  89.7  51.1  21.1
##  9  72.6  9.55  35.6  90.5  50.9  58.9  24.6  86.1  37.2  47.0  87.7  51.8  21.4
## 10  75.4  9.16  35.6  92.4  50.1  60.4  25.5  86.3  37.7  47.5  86.1  53.2  21.0
## # i 90 more rows
## # i 12 more variables: ...14 <dbl>, ...15 <dbl>, ...16 <dbl>, ...17 <dbl>,
## #   ...18 <dbl>, ...19 <dbl>, ...20 <dbl>, ...21 <dbl>, ...22 <dbl>,
## #   ...23 <dbl>, ...24 <dbl>, ...25 <dbl>
##
## $Volume
## # A tibble: 100 x 25
##      ...1   ...2   ...3   ...4   ...5   ...6   ...7   ...8   ...9  ...10  ...11
##     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 396300 575800 484300 703700 685000  72100 432300 530700 268100 606700  14700
##  2 688400 888900 127200 308400 573200 478300 530600 580100 442800 739800 496400
##   3 434800 283400 498600 381300 225800 921800 157300  81000 788300 660000 720700
##   4 109900 162100 837800 635100 469000 383900 146400 529600 204800 635200 939100
##   5 467000 190300 631200  48600 740500  52400 189600 567300 382400 316800 101700
##   6 119300 597000  74600 808200 618100 591100  92400 904400 452000  97500  97200
##   7 812300 170000 671400 420000 282400  76300 272300 660700 650500 778100 526700
##   8 579900 468700 904100 368100 360600 129400 781500 579200 167000 161200 918800
##   9  86800 505400 831200 573100 199700 835000 626700 837700 344200  89500 578500
##  10 564900 217000 263300 628000 696100 947100 542400  77300 805700 526800 950500
## # i 90 more rows
## # i 14 more variables: ...12 <dbl>, ...13 <dbl>, ...14 <dbl>, ...15 <dbl>,
## #   ...16 <dbl>, ...17 <dbl>, ...18 <dbl>, ...19 <dbl>, ...20 <dbl>,
## #   ...21 <dbl>, ...22 <dbl>, ...23 <dbl>, ...24 <dbl>, ...25 <dbl>
##
## $Parameters
## # A tibble: 11 x 4
##    ...1          ...2 ...3  ...4
##    <chr>        <dbl> <lgl> <chr>
##  1 WeightHigh1    0.1 NA    These are the weights we will place on the H/L/O/C ~
##  2 WeightLow1     0.1 NA    Day 1 is the current trading day and Day 2 is the p~
##  3 WeightOpen1    0.1 NA    We can't calculate this value until Day 2 of the Sa~
##  4 WeightClose1   0.2 NA    <NA>
##  5 WeightHigh2    0.1 NA    <NA>
##  6 WeightLow2     0.1 NA    <NA>
##  7 WeightOpen2    0.1 NA    <NA>
##  8 WeightLow2     0.2 NA    <NA>
##  9 <NA>          NA   NA    <NA>
## 10 Buy Max       -1   NA    These are the signals we require to initiate buy an~
## 11 Sell Min       1   NA    This particular strategy buys stocks after they go ~
```

```r
# Creating the parameters data frame
parameters <- data.frame(
  Parameter = c("WeightHigh1", "WeightLow1", "WeightOpen1", "WeightClose1",
                "WeightHigh2", "WeightLow2", "WeightOpen2", "WeightClose2",
                "BuyMax", "SellMin"),
  Value = c(0.1, 0.1, 0.1, 0.2, 0.1, 0.1, 0.1, 0.2, -1, 1)
)

# Display the data frame
head(parameters)
```

```
##      Parameter Value
## 1  WeightHigh1   0.1
## 2   WeightLow1   0.1
## 3  WeightOpen1   0.1
## 4 WeightClose1   0.2
## 5  WeightHigh2   0.1
## 6   WeightLow2   0.1
```

```r
# Initialize a blank data frame for the result
price <- as.data.frame(matrix(0, nrow=nrow(data$High), ncol=ncol(data$High)))
colnames(price) <- colnames(data$High)

# Copy the first row of data$High to the first row of price
```

```
price[1, ] <- data$High[1, ]

# Perform the calculation for each row, starting from the second row
for (i in 2:nrow(data$High)) {
  price[i, ] <- parameters$Value[parameters$Parameter == "WeightHigh1"] * data$High[i, ] +
                parameters$Value[parameters$Parameter == "WeightLow1"] * data$Low[i, ] +
                parameters$Value[parameters$Parameter == "WeightOpen1"] * data$Open[i, ] +
                parameters$Value[parameters$Parameter == "WeightClose1"] * data$Close[i, ] +
                parameters$Value[parameters$Parameter == "WeightHigh2"] * data$High[i-1, ] +
                parameters$Value[parameters$Parameter == "WeightLow2"] * data$Low[i-1, ] +
                parameters$Value[parameters$Parameter == "WeightOpen2"] * data$Open[i-1, ] +
                parameters$Value[parameters$Parameter == "WeightClose2"] * data$Close[i-1, ]
}

# Set the first row to NA to indicate it's intentionally left empty
price[1, ] <- NA


# Display the resulting price data frame
head(price)
```

```
##        ...1     ...2     ...3     ...4     ...5     ...6     ...7     ...8
## 1        NA       NA       NA       NA       NA       NA       NA       NA
## 2 73.27874 11.27244 33.93538 86.23187 46.45375 54.34495 26.36801 79.00297
## 3 72.28340 10.93532 34.75323 83.79944 48.09550 53.64132 26.20424 81.70979
## 4 72.86426 10.83068 35.35273 83.94984 49.41745 51.22524 25.91902 81.68839
## 5 72.42985 11.10792 36.02966 86.94852 50.37168 51.27151 25.23505 80.31576
## 6 70.45999 10.59292 36.09669 88.83884 50.74981 53.36907 25.18481 80.24356
##        ...9    ...10    ...11    ...12    ...13    ...14    ...15    ...16
## 1        NA       NA       NA       NA       NA       NA       NA       NA
## 2 34.47829 47.05930 87.37693 48.70624 18.95886 11.66296 26.21579 32.63184
## 3 34.74171 47.32061 90.23727 50.68676 19.45311 11.90775 26.14488 33.03164
## 4 34.72910 46.42574 92.32358 51.61702 19.84646 12.17971 26.48996 32.23173
## 5 33.95614 45.59495 91.73579 52.34123 19.84799 12.53243 27.22323 31.75972
## 6 34.06700 46.55826 91.07229 52.02908 20.61236 12.70770 27.50681 32.63185
##       ...17    ...18    ...19    ...20    ...21    ...22    ...23    ...24
## 1        NA       NA       NA       NA       NA       NA       NA       NA
## 2 48.84419 73.08078 68.43565 49.86104 81.89058 93.78361 52.87055 48.08784
## 3 49.59667 73.42547 70.21340 49.30180 81.65604 93.50981 53.28951 47.34360
## 4 48.63363 74.46216 71.95896 49.75266 81.65311 91.06616 53.11432 47.24624
## 5 47.39980 73.61401 74.25104 49.53989 83.45982 91.44016 51.68549 48.08156
## 6 47.69330 73.80102 75.15080 48.58895 86.66038 93.40172 49.31936 47.75902
##       ...25
## 1        NA
## 2 38.67948
## 3 39.43509
## 4 39.08575
## 5 38.51500
## 6 38.10636
```

```
#return
# Initialize a blank data frame for the result
return <- as.data.frame(matrix(0, nrow=nrow(data$High)-1, ncol=ncol(data$High)))
```

```
colnames(return) <- colnames(data$High)

# Perform the calculation for each row
for (i in 3:nrow(data$High)) {
  return[i, ] <- 100 * (((data$Close[i, ])-price[i-1, ])/price[i-1, ])
}

# Display the resulting price data frame
head(return)
```

```
##        ...1      ...2      ...3      ...4        ...5      ...6        ...7
## 1  0.000000  0.000000  0.000000  0.000000  0.00000000  0.000000  0.00000000
## 2  0.000000  0.000000  0.000000  0.000000  0.00000000  0.000000  0.00000000
## 3 -1.818721 -3.382465  3.196926 -5.181998  4.78006951 -3.219393 -0.06369828
## 4  2.675779  1.001330  2.278368  2.257485  3.05309647 -6.372303 -1.84404742
## 5 -2.777291 -1.637402  1.543661  4.217407  2.64814594  1.927524 -3.40758743
## 6 -3.087999 -8.444788 -1.760233  2.973012 -0.09747642  6.153397 -0.57227225
##         ...8      ...9      ...10      ...11     ...12    ...13      ...14
## 1  0.0000000  0.000000  0.0000000  0.0000000  0.000000 0.000000  0.0000000
## 2  0.0000000  0.000000  0.0000000  0.0000000  0.000000 0.000000  0.0000000
## 3  4.4912627  3.237506 -0.8794155  4.7065876  5.584565 3.404139  0.8230380
## 4 -0.8048472 -2.538550 -2.4373261  3.6626888  1.434713 1.250500  2.3296454
## 5 -1.7495283 -3.487431 -1.4880733 -2.7617701  1.980689 1.768643 -0.3181262
## 6  0.2213636  0.296681  4.7119060  0.2129392 -2.335398 6.093557 -0.8189741
##       ...15      ...16     ...17      ...18      ...19     ...20      ...21
## 1 0.0000000  0.0000000  0.000000  0.0000000  0.0000000  0.000000  0.0000000
## 2 0.0000000  0.0000000  0.000000  0.0000000  0.0000000  0.000000  0.0000000
## 3 0.1870276  1.0929363  1.314300  1.6181967  3.6367382 -1.860608 -1.1040583
## 4 5.2993366 -5.0968251 -3.698216  1.4679892  3.5651431  1.130405  0.2716363
## 5 2.4075957 -0.1820841 -3.973867 -1.8428247  4.9738442 -1.253283  4.4373431
## 6 2.2877878  3.9848287  1.945799  0.7892446 -0.4093902 -2.788741  5.6026842
##        ...22      ...23      ...24     ...25
## 1  0.0000000  0.0000000  0.0000000  0.000000
## 2  0.0000000  0.0000000  0.0000000  0.000000
## 3 -0.8336931  1.4932491 -2.7885035  3.617700
## 4 -4.1039004 -0.8975026  0.3427454 -2.347643
## 5  2.2669957 -5.4541586  2.4476470 -1.378015
## 6  1.8890798 -6.0979990 -2.3491326 -3.961954
```

```
#signal
# Initialize a blank data frame for the result
signal <- as.data.frame(matrix(0, nrow=nrow(data$High)-1, ncol=ncol(data$High)))
colnames(signal) <- colnames(data$High)

# Perform the calculation for each row
for (i in 3:nrow(data$High)) {
  signal[i, ] <- (10000000 * return[i,])/(data$Close[i, ]*data$Volume[i, ])
}

# Display the resulting price data frame
head(signal)
```

```
##         ...1       ...2       ...3       ...4        ...5      ...6       ...7
## 1  0.0000000   0.000000  0.0000000  0.0000000  0.00000000  0.000000  0.0000000
## 2  0.0000000   0.000000  0.0000000  0.0000000  0.00000000  0.000000  0.0000000
## 3 -0.5813931 -10.958710  1.8308842 -1.6621560  4.34921392 -0.664033 -0.1536733
## 4  3.2805446   5.592885  0.7650760  0.4148075  1.31341555 -3.305023 -4.8971426
## 5 -0.8395029  -8.076644  0.6812542  9.9185704  0.70499392  7.045195 -7.1787204
## 6 -3.6875808 -13.909091 -6.6662865  0.4108584 -0.03133848  1.912687 -2.4684199
##           ...8       ...9      ...10      ...11      ...12      ...13     ...14
## 1  0.00000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.000000
## 2  0.00000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.000000
## 3  6.71676335  1.1538141 -0.2856544  0.7138070  2.0065337  4.3443324  1.327879
## 4 -0.18749986 -3.6607636 -0.8311302  0.4169460  0.3503908  0.6785174  2.868524
## 5 -0.38424941 -2.7208859 -1.0270497 -3.0249408  0.5514804  1.3308141 -7.101006
## 6  0.03040777  0.1927287 10.1223004  0.2383015 -2.0179172 12.9128877 -2.793048
##         ...15       ...16      ...17      ...18       ...19       ...20
## 1 0.0000000  0.00000000  0.0000000  0.0000000  0.00000000   0.0000000
## 2 0.0000000  0.00000000  0.0000000  0.0000000  0.00000000   0.0000000
## 3 0.0716166  0.49126403  0.2932099  0.2321045  0.73874380  -0.8917291
## 4 2.0313470 -3.68012981 -1.4075498  0.2073197  1.25648144   0.2455538
## 5 9.0010494 -0.07854997 -0.9999026 -0.2789369  3.16869652  -0.2720775
## 6 0.8216667  1.33370089  0.5187075  0.2810419 -0.06250012 -21.2896087
##          ...21      ...22      ...23       ...24         ...25
## 1  0.0000000  0.0000000  0.0000000  0.00000000     0.0000000
## 2  0.0000000  0.0000000  0.0000000  0.00000000     0.0000000
## 3 -2.1917408 -0.1413255  0.5376344 -6.35261736    17.3585965
## 4  0.1104387 -0.5818148 -0.3293515  0.07297261    -1.7427970
## 5  1.8023855  0.8541091 -2.6177625  1.32170514    -0.6381438
## 6  1.1142640  2.1478981 -2.0747127 -1.52538355 -1785.1921412
```

```r
#closeclosereturn
# Initialize a blank data frame for the result
close_close_return <- as.data.frame(matrix(0, nrow=nrow(data$High)-1, ncol=ncol(data$High)))
colnames(close_close_return) <- colnames(data$High)

# Perform the calculation for each row
for (i in 2:nrow(data$High)) {
  close_close_return[i, ] <- 100*((data$Close[i, ]-data$Close[i-1, ])/data$Close[i-1, ])
}

# Display the resulting price data frame
head(close_close_return)
```

```
##           ...1      ...2       ...3      ...4      ...5      ...6       ...7
## 1  0.0000000  0.000000  0.0000000  0.000000  0.000000  0.000000  0.0000000
## 2 -0.6163136 -4.605988  3.6366810 -1.189022  2.999041  1.906567 -1.7251899
## 3 -1.4810760 -2.010423  1.4170055 -4.482628  3.853567 -3.982723  0.6590230
## 4  3.1572880  1.410879  1.4984642  4.803935  1.827721 -4.510370 -2.3915115
## 5 -4.5500466 -3.544483  0.9942941  2.099571  2.344840  3.961302 -2.6640264
## 6 -0.9138784 -4.538111 -1.4011955  2.335306 -0.795489  4.240027  0.2189938
##           ...8      ...9      ...10     ...11       ...12      ...13      ...14
## 1  0.0000000  0.000000  0.0000000  0.000000  0.00000000  0.0000000  0.0000000
## 2  4.4407923 -1.153498  2.6148877  3.682692  3.67004343 -0.7890681  1.4478915
## 3  2.4988281  3.804727 -1.9638438  2.920418  2.61645102  4.8445027 -2.8475296
## 4 -1.8159034 -4.873629 -1.0251665  2.243944 -0.02392613  0.4699330  3.6245592
## 5 -0.9782880 -1.009556 -0.9365105 -4.028721  2.38343210  2.5441307 -0.3626813
## 6  0.2919594  1.607912  4.3914904  2.403058 -2.88859322  4.2578168  2.3789547
##        ...15     ...16      ...17      ...18     ...19      ...20      ...21
## 1  0.0000000  0.000000  0.0000000  0.0000000  0.000000  0.0000000  0.0000000
## 2  3.5579868  4.230399  3.1938800 -2.8877128  1.566951 -1.7497735 -0.7954254
## 3 -0.8865535 -0.885019  0.1040538  3.0888236  3.180423 -0.7806684 -0.4631564
## 4  4.8184729 -4.972666 -3.4831313  0.3231428  2.526817  1.8919503  1.1006582
## 5 -1.4625845  2.631621 -2.2224202 -1.8970970  3.880102 -1.4641192  4.1506829
## 6  2.6478817  2.648962  3.4712537  1.5119041 -2.106251 -1.9759431  3.3531831
##         ...22      ...23     ...24       ...25
## 1  0.00000000  0.0000000  0.000000  0.00000000
## 2  0.80599856  0.5341021 -1.958053  2.39092707
## 3 -1.18443197  1.1636336 -1.964833  2.81211350
## 4 -3.58002033 -1.5818087  1.623543 -3.91602557
## 5  3.85666945 -4.9115543  1.887758  0.09829276
## 6  0.03964052 -3.3527651 -2.996951 -4.04205190
```

```r
#Buy
# Initialize a blank data frame for the result
buy <- as.data.frame(matrix(0, nrow=nrow(data$High), ncol=ncol(data$High)))
colnames(buy) <- colnames(data$High)

# Perform the calculation for each row
for (i in 3:nrow(data$High)) {
  buy[i, ] <- ifelse(signal[i, ] < parameters$Value[parameters$Parameter == "BuyMax"], 1, 0)
}

head(buy)
```

```
##   ...1 ...2 ...3 ...4 ...5 ...6 ...7 ...8 ...9 ...10 ...11 ...12 ...13 ...14
## 1    0    0    0    0    0    0    0    0    0     0     0     0     0     0
## 2    0    0    0    0    0    0    0    0    0     0     0     0     0     0
## 3    0    1    0    1    0    0    0    0    0     0     0     0     0     0
## 4    0    0    0    0    0    1    1    0    1     0     0     0     0     0
## 5    0    1    0    0    0    0    1    0    1     1     1     0     0     1
## 6    1    1    1    0    0    0    1    0    0     0     0     1     0     1
##   ...15 ...16 ...17 ...18 ...19 ...20 ...21 ...22 ...23 ...24 ...25
## 1     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0     0     0     0     0
## 3     0     0     0     0     0     0     1     0     0     1     0
## 4     0     1     1     0     0     0     0     0     0     0     1
## 5     0     0     0     0     0     0     0     0     1     0     0
## 6     0     0     0     0     0     1     0     0     1     1     1
```

```r
#Sell
# Initialize a blank data frame for the result
sell <- as.data.frame(matrix(0, nrow=nrow(data$High), ncol=ncol(data$High)))
colnames(sell) <- colnames(data$High)

# Perform the calculation for each row
for (i in 3:nrow(data$High)) {
  sell[i, ] <- ifelse(signal[i, ] > parameters$Value[parameters$Parameter == "SellMin"], 1, 0)
}

head(sell)
```

```
##   ...1 ...2 ...3 ...4 ...5 ...6 ...7 ...8 ...9 ...10 ...11 ...12 ...13 ...14
## 1    0    0    0    0    0    0    0    0    0     0     0     0     0     0
## 2    0    0    0    0    0    0    0    0    0     0     0     0     0     0
## 3    0    0    1    0    1    0    0    1    1     0     0     1     1     1
## 4    1    1    0    0    1    0    0    0    0     0     0     0     0     1
## 5    0    0    0    1    0    1    0    0    0     0     0     0     1     0
## 6    0    0    0    0    0    1    0    0    0     1     0     0     1     0
##   ...15 ...16 ...17 ...18 ...19 ...20 ...21 ...22 ...23 ...24 ...25
## 1     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0     0     0     0     0
## 3     0     0     0     0     0     0     0     0     0     0     1
## 4     1     0     0     0     1     0     0     0     0     0     0
## 5     1     0     0     0     1     0     1     0     0     1     0
## 6     0     1     0     0     0     0     1     1     0     0     0
```

```
# Assuming 'buy' and 'close_close_return' are data frames with the same dimensions

# Initialize the PNL data frame with a Long_Return column
PNL <- data.frame(Long_Return = numeric(nrow(buy)),
                  Short_Return = numeric(nrow(buy)),
                  Strategy_Return = numeric(nrow(buy)))

# Perform the calculation for each row starting from the 3rd row
for (i in 3:nrow(buy)) {
  PNL$Long_Return[i] <- sum(buy[i, ] * close_close_return[i+1, ])/sum(buy[i, ])
  PNL$Short_Return[i] <- sum(sell[i, ] * close_close_return[i+1, ])/sum(sell[i, ])
  PNL$Strategy_Return[i] <- PNL$Long_Return[i] - PNL$Short_Return[i]
}


# Remove the first two and the last row from the PNL data frame
PNL <- PNL[-c(1, 2, nrow(PNL)), ]

# Display the resulting PNL data frame
head(PNL)
```

```
##   Long_Return Short_Return Strategy_Return
## 3   2.2347537   -0.4011007       2.6358545
## 4   0.1325355   -0.6158090       0.7483445
## 5   0.4442190    1.6758588      -1.2316398
## 6  -1.3000981    0.5194445      -1.8195425
## 7   0.8525412    1.1799185      -0.3273773
## 8   2.5837770    1.2479824       1.3357946
```

```
# Assuming PNL$Strategy_Return is already defined

# Calculate the Sharpe ratio
Sharpe_ratio <- 16 * mean(PNL$Strategy_Return, na.rm = TRUE) / sd(PNL$Strategy_Return, na.rm = TRUE)


# Print the average daily return
cat("average daily return:", mean(PNL$Strategy_Return, na.rm=TRUE), "\n")
```

```
## average daily return: 0.1794042
```

```
# Print the st deviation
cat("st deviation is:", sd(PNL$Strategy_Return, na.rm=TRUE), "\n")
```

```
## st deviation is: 1.665773
```

```
# Print the Sharpe ratio
cat("Sharpe Ratio is:", Sharpe_ratio, "\n")
```

```
## Sharpe Ratio is: 1.723204
```

```
# Print the Annual win probability
cat("Annual Win probability:", pnorm(Sharpe_ratio), "\n")
```

```
## Annual Win probability: 0.9575742
```