Vectorized code for stocks data analysis

Aayush

2024-06-24

```
library(pacman)
pacman::p_load(tidyverse, readxl)

# File paths
file_path_equity <- "C:\\Users\\Aayush\\Documents\\dr. moore stock project\\excel files\\DailyDataV1.csv
file_path_no_equity <- "C:\\Users\\Aayush\\Documents\\dr. moore stock project\\excel files\\fakedata_no_

# Read data
data_equity <- read_csv(file_path_equity, col_names = FALSE)
```

```
## Rows: 1324 Columns: 14641
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (14641): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X1...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_no_equity <- read_excel(file_path_no_equity)
```

```
## New names:
## * `OPEN` -> `OPEN...2`
## * `LAST_PRICE` -> `LAST_PRICE...3`
## * `PX_HIGH_ALL_SESSION` -> `PX_HIGH_ALL_SESSION...4`
## * `PX_LOW_ALL_SESSION` -> `PX_LOW_ALL_SESSION...5`
## * `VOLUME` -> `VOLUME...6`
## * `OPEN` -> `OPEN...7`
## * `LAST_PRICE` -> `LAST_PRICE...8`
## * `PX_HIGH_ALL_SESSION` -> `PX_HIGH_ALL_SESSION...9`
## * `PX_LOW_ALL_SESSION` -> `PX_LOW_ALL_SESSION...10`
## * `VOLUME` -> `VOLUME...11`
## * `OPEN` -> `OPEN...12`
## * `LAST_PRICE` -> `LAST_PRICE...13`
## * `PX_HIGH_ALL_SESSION` -> `PX_HIGH_ALL_SESSION...14`
## * `PX_LOW_ALL_SESSION` -> `PX_LOW_ALL_SESSION...15`
## * `VOLUME` -> `VOLUME...16`
## * `OPEN` -> `OPEN...17`
## * `LAST_PRICE` -> `LAST_PRICE...18`
## * `PX_HIGH_ALL_SESSION` -> `PX_HIGH_ALL_SESSION...19`
## * `PX_LOW_ALL_SESSION` -> `PX_LOW_ALL_SESSION...20`
## * `VOLUME` -> `VOLUME...21`
```

```r
# Extract equity names
equity_names <- as.character(data_equity[1, -1])
equity_names <- na.omit(equity_names)

# Prepare Dates and data
dates <- data_no_equity[, 1]
data_equity <- data_equity[-c(1, 2), -1]

# Initialize dataframes
open_df <- data.frame(Dates = dates)
last_price_df <- data.frame(Dates = dates)
px_high_df <- data.frame(Dates = dates)
px_low_df <- data.frame(Dates = dates)
volume_df <- data.frame(Dates = dates)

# Number of columns per equity
columns_per_equity <- 5

# Loop through each equity and create separate dataframes
for (i in 1:length(equity_names)) {
  start_col <- (i - 1) * columns_per_equity + 1
  end_col <- start_col + columns_per_equity - 1

  equity_data <- data_equity[, start_col:end_col]
  equity_data <- cbind(dates, equity_data)
  colnames(equity_data) <- c("Dates", "OPEN", "LAST_PRICE", "PX_HIGH_ALL_SESSION",
                             "PX_LOW_ALL_SESSION", "VOLUME")

  # Convert relevant columns to numeric
  equity_data$OPEN <- as.numeric(equity_data$OPEN)
  equity_data$LAST_PRICE <- as.numeric(equity_data$LAST_PRICE)
  equity_data$PX_HIGH_ALL_SESSION <- as.numeric(equity_data$PX_HIGH_ALL_SESSION)
  equity_data$PX_LOW_ALL_SESSION <- as.numeric(equity_data$PX_LOW_ALL_SESSION)
  equity_data$VOLUME <- as.numeric(equity_data$VOLUME)



  # Ensure there are no missing values in the subsetting operations
  # Replace rows where Volume is less than or equal to 0 with NA (excluding Dates)
  equity_data[!is.na(equity_data$VOLUME) & equity_data$VOLUME <= 0, 2:6] <- NA

  # Replace rows where PX_HIGH_ALL_SESSION is less than PX_LOW_ALL_SESSION with NA (excluding Dates)
  equity_data[!is.na(equity_data$PX_HIGH_ALL_SESSION) & !is.na(equity_data$PX_LOW_ALL_SESSION) &
                equity_data$PX_HIGH_ALL_SESSION < equity_data$PX_LOW_ALL_SESSION, 2:6] <- NA

  # Replace rows where PX_HIGH_ALL_SESSION / LAST_PRICE > 1.9 with NA (excluding Dates)
  equity_data[!is.na(equity_data$PX_HIGH_ALL_SESSION) & !is.na(equity_data$LAST_PRICE) &
                (equity_data$PX_HIGH_ALL_SESSION / equity_data$LAST_PRICE) > 1.9, 2:6] <- NA

  # Replace rows where LAST_PRICE / PX_LOW_ALL_SESSION < 1.9 with NA (excluding Dates)
  equity_data[!is.na(equity_data$LAST_PRICE) & !is.na(equity_data$PX_LOW_ALL_SESSION) &
                (equity_data$LAST_PRICE / equity_data$PX_LOW_ALL_SESSION) > 1.9, 2:6] <- NA
```

```
  # Add the relevant columns to the respective dataframes
  open_df[[equity_names[i]]] <- equity_data$OPEN
  last_price_df[[equity_names[i]]] <- equity_data$LAST_PRICE
  px_high_df[[equity_names[i]]] <- equity_data$PX_HIGH_ALL_SESSION
  px_low_df[[equity_names[i]]] <- equity_data$PX_LOW_ALL_SESSION
  volume_df[[equity_names[i]]] <- equity_data$VOLUME
}

# # Remove column names
# colnames(open_df) <- NULL
# colnames(last_price_df) <- NULL
# colnames(px_high_df) <- NULL
# colnames(px_low_df) <- NULL
# colnames(volume_df) <- NULL

# Remove Dates columns
open_df <- open_df[, -1]
last_price_df <- last_price_df[, -1]
px_high_df <- px_high_df[, -1]
px_low_df <- px_low_df[, -1]
volume_df <- volume_df[, -1]


# Display the first few rows of each dataframe
open_df[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1        21.05        114.56        54.70           3.63       338.19
## 2        21.67        115.86        56.62           3.75       342.57
## 3        22.02        117.18        56.55           3.73       346.49
## 4        21.45        118.42        57.11           3.58       348.18
## 5        21.53        119.82        58.00           3.60       353.10
```

```
last_price_df[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1        21.51        114.62        56.38           3.69       338.89
## 2        21.89        116.85        56.22           3.72       344.62
## 3        21.37        118.12        57.03           3.58       348.75
## 4        21.57        119.43        57.54           3.59       350.64
## 5        21.44        121.11        57.24           3.49       353.70
```

```
px_high_df[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1        21.66        115.90       56.490           3.74       339.10
## 2        22.23        116.99       56.975           3.86       346.39
## 3        22.02        118.23       57.370           3.77       353.10
## 4        21.98        119.64       57.810           3.61       351.63
## 5        21.63        121.35       58.130           3.64       355.25
```

```r
px_low_df[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1        20.83        113.90       54.560          3.590       330.68
## 2        21.48        115.24       55.925          3.700       340.12
## 3        21.04        117.02       56.500          3.560       345.02
## 4        21.23        118.15       57.100          3.515       345.60
## 5        20.92        119.65       57.150          3.470       352.50
```

```r
volume_df[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1       702098        849381      4534778         112818       901720
## 2       679581        817892      3987577          70370       595357
## 3       735204        518875      2977202          83821       561686
## 4       564600        822715      2654552          70397       488016
## 5       617387        545070      3105705          68403       463831
```

```r
# Creating the parameters data frame
parameters <- data.frame(
  Parameter = c("WeightHigh1", "WeightLow1", "WeightOpen1", "WeightClose1",
                "WeightHigh2", "WeightLow2", "WeightOpen2", "WeightClose2",
                "BuyMax", "SellMin"),
  Value = c(1, 0, 0, 0, 0, 0, 0, 0, -1, 1)
)

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

```
##      Parameter Value
## 1  WeightHigh1     1
## 2   WeightLow1     0
## 3  WeightOpen1     0
## 4 WeightClose1     0
## 5  WeightHigh2     0
## 6   WeightLow2     0
```

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

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

# Retrieve parameter values
w_high1 <- parameters$Value[parameters$Parameter == "WeightHigh1"]
w_low1 <- parameters$Value[parameters$Parameter == "WeightLow1"]
w_open1 <- parameters$Value[parameters$Parameter == "WeightOpen1"]
w_close1 <- parameters$Value[parameters$Parameter == "WeightClose1"]
w_high2 <- parameters$Value[parameters$Parameter == "WeightHigh2"]
w_low2 <- parameters$Value[parameters$Parameter == "WeightLow2"]
```

```
w_open2 <- parameters$Value[parameters$Parameter == "WeightOpen2"]
w_close2 <- parameters$Value[parameters$Parameter == "WeightClose2"]

# Perform the calculation using vectorized operations
price[-1, ] <- w_high1 * px_high_df[-1, ] +
               w_low1 * px_low_df[-1, ] +
               w_open1 * open_df[-1, ] +
               w_close1 * last_price_df[-1, ] +
               w_high2 * px_high_df[-nrow(px_high_df), ] +
               w_low2 * px_low_df[-nrow(px_low_df), ] +
               w_open2 * open_df[-nrow(open_df), ] +
               w_close2 * last_price_df[-nrow(last_price_df), ]

# Display the resulting price data frame
price[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1           NA            NA           NA             NA           NA
## 2        22.23        116.99       56.975           3.86       346.39
## 3        22.02        118.23       57.370           3.77       353.10
## 4        21.98        119.64       57.810           3.61       351.63
## 5        21.63        121.35       58.130           3.64       355.25
```

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

# Perform the calculation using vectorized operations
return[-1, ] <- 100 * ((last_price_df[-(1:2), ] - price[-c(1, nrow(price)), ]) /
                       price[-c(1, nrow(price)),  ])

#add 0 row
return <- rbind(0, return)

# Display the resulting return data frame
return[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1     0.000000     0.0000000   0.00000000       0.000000    0.0000000
## 2     0.000000     0.0000000   0.00000000       0.000000    0.0000000
## 3    -3.868646     0.9658945   0.09653357      -7.253886    0.6813130
## 4    -2.043597     1.0149708   0.29632212      -4.774536   -0.6966865
## 5    -2.456779     1.2286861  -0.98598858      -3.324100    0.5886870
```

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

# Perform the calculation using vectorized operations
signal[-(1:2), ] <- (10000000 * return[-(1:2), ]) / (last_price_df[-(1:2), ] *
                                                     volume_df[-(1:2), ])
```

```r
#replace Inf with NA
signal[sapply(signal, is.infinite)] <- NA

# Display the resulting signal data frame
signal[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1     0.000000     0.0000000  0.000000000         0.0000   0.00000000
## 2     0.000000     0.0000000  0.000000000         0.0000   0.00000000
## 3    -2.462332     0.1575954  0.005685474      -241.7324   0.03478074
## 4    -1.678047     0.1032977  0.019400056      -188.9220  -0.04071382
## 5    -1.856025     0.1861267 -0.055464111      -139.2430   0.03588307
```

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

# Handle the first row by setting it to NA (or some other initial value if needed)
close_close_return[1, ] <- NA

# Perform the calculation using vectorized operations
close_close_return[-1, ] <- 100 * ((last_price_df[-1, ] -
                                    last_price_df[-nrow(last_price_df), ]) / last_price_df[-nrow(last

# Display the resulting close_close_return data frame
close_close_return[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1           NA           NA           NA             NA           NA
## 2    1.7666202     1.945559   -0.2837886      0.8130081    1.6908141
## 3   -2.3755139     1.086864    1.4407684     -3.7634409    1.1984214
## 4    0.9358914     1.109042    0.8942662      0.2793296    0.5419355
## 5   -0.6026889     1.406682   -0.5213764     -2.7855153    0.8726899
```

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

# Retrieve the BuyMax parameter value
buy_max <- as.numeric(parameters$Value[parameters$Parameter == "BuyMax"])

# Perform the calculation using vectorized operations
buy[3:nrow(px_high_df), ] <- ifelse(signal[3:nrow(px_high_df), ] < buy_max, 1, 0)



buy[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1            0            0            0              0            0
## 2            0            0            0              0            0
## 3            1            0            0              1            0
```

```
## 4            1             0            0             1            0
## 5            1             0            0             1            0
```

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

# Retrieve the SellMin parameter value
sell_min <- as.numeric(parameters$Value[parameters$Parameter == "SellMin"])

# Perform the calculation using vectorized operations
sell[3:nrow(px_high_df), ] <- ifelse(signal[3:nrow(px_high_df), ] > sell_min, 1, 0)




sell[1:5,1:5]
```

```
##   AA UN Equity AXP UN Equity VZ UN Equity SPWH UW Equity BA UN Equity
## 1            0             0            0             0            0
## 2            0             0            0             0            0
## 3            0             0            0             0            0
## 4            0             0            0             0            0
## 5            0             0            0             0            0
```

```
# 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 using vectorized operations, starting from the 3rd row
buy_sum <- rowSums(buy[3:nrow(buy), ],na.rm = TRUE)
sell_sum <- rowSums(sell[3:nrow(buy), ], na.rm=TRUE)

# # Avoid division by zero by replacing zeros in buy_sum and sell_sum with NA
# buy_sum[buy_sum == 0] <- NA
# sell_sum[sell_sum == 0] <- NA

PNL$Long_Return[3:(nrow(buy) - 1)] <- rowSums(buy[3:(nrow(buy) - 1), ] *
                                      close_close_return[4:nrow(close_close_return), ], na.rm = TRUE) /
```

```
## Warning in rowSums(buy[3:(nrow(buy) - 1), ] *
## close_close_return[4:nrow(close_close_return), : longer object length is not a
## multiple of shorter object length
```

```
## Warning in PNL$Long_Return[3:(nrow(buy) - 1)] <- rowSums(buy[3:(nrow(buy) - :
## number of items to replace is not a multiple of replacement length
```

```
PNL$Short_Return[3:(nrow(buy) - 1)] <- rowSums(sell[3:(nrow(buy) - 1), ] *
                                       close_close_return[4:nrow(close_close_return), ],na.rm = TRUE) /
```

```
## Warning in rowSums(sell[3:(nrow(buy) - 1), ] *
## close_close_return[4:nrow(close_close_return), : longer object length is not a
## multiple of shorter object length
```

```
## Warning in PNL$Short_Return[3:(nrow(buy) - 1)] <- rowSums(sell[3:(nrow(buy) - :
## number of items to replace is not a multiple of replacement length
```

```
PNL$Strategy_Return[3:(nrow(buy) - 1)] <- PNL$Long_Return[3:(nrow(buy) - 1)] - PNL$Short_Return[3:(nrow

# 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  -0.6086789   -0.1467390     -0.46193996
## 4   0.7629763    0.8571403     -0.09416393
## 5   0.5104953    0.8905713     -0.38007594
## 6  -0.2699448   -0.2448307     -0.02511418
## 7  -0.0116363   -0.5023148      0.49067846
## 8   1.3646149    0.8632574      0.50135745
```

```
# 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: 0.0218613
```

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

```
## st deviation is: 0.8539584
```

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

```
## Sharpe Ratio is: 0.4095994
```

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

```
## Annual Win probability: 0.6589501
```